Citation B1
173 Pages

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 December 2003 (11.12.2003)

PCT

(10) International Publication Number
WO 03/102854 A2

(51) International Patent Classification[7]: G06F 19/00

(21) International Application Number: PCT/US03/17940

(22) International Filing Date: 4 June 2003 (04.06.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/386,296 4 June 2002 (04.06.2002) US
60/411,574 16 September 2002 (16.09.2002) US

(71) Applicant: APPLERA CORPORATION [US/US]; 850 Lincoln Centre, Foster City, CA 94404 (US).

(72) Inventors: ROHRLICH, John; Santa Fe, NM (US). YUNG, Kai; Foster City, CA (US). FANG, Sylvia; San Jose, CA (US). DODGEN, Steve; Foster City, CA (US).

(74) Agent: HUNT, Dale, C.; KNOBBE, MARTENS, OLSON & BEAR, LLP, 2040 Main Street, 14th Floor, Irvine, CA 92614 (US).

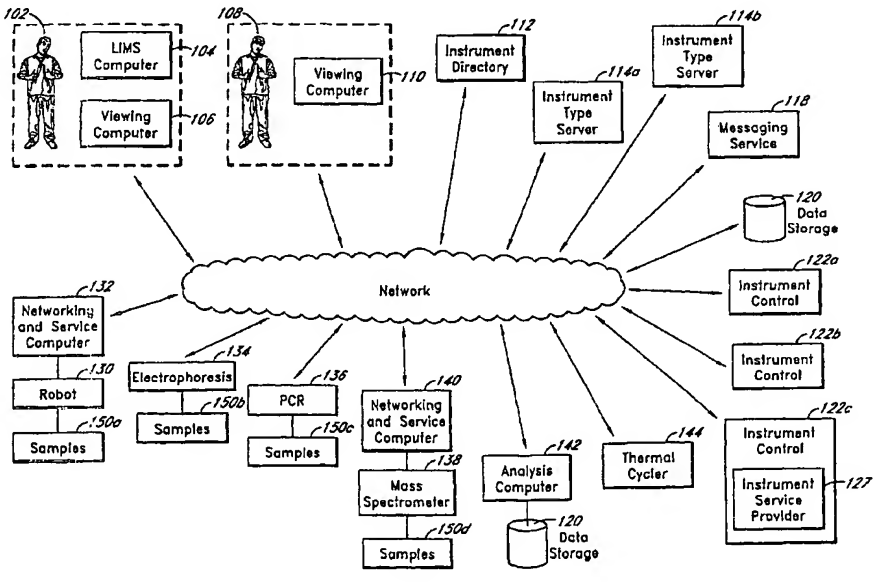
(81) Designated States *(national)*: AE, AG, AL, AM, AT (utility model), AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ (utility model), CZ, DE (utility model), DE, DK (utility model), DK, DM, DZ, EC, EE (utility model), EE, ES, FI (utility model), FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK (utility model), SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR OPEN CONTROL AND MONITORING OF BIOLOGICAL INSTRUMENTS



(57) Abstract: A system and methods for integrating laboratory instrumentation and applications to provide a unified control and coordination architecture under a common interface. The system may be adapted to a variety of different hardware and software components wherein the individual functionalities and input/output data types for each component are recognized and incorporated into a centralized control and monitoring system.

WO 03/102854 A2

SYSTEM AND METHOD FOR OPEN CONTROL AND MONITORING OF BIOLOGICAL INSTRUMENTS

Background

Field

The present invention relates to computer-controlled biological instrumentation, and in particular to controlling information flow to and from biological instruments.

Description of the Related Art

Biological laboratories, in particular those focused on genetic and molecular biological work, employ a wide variety of instrumentation in order to perform experiments and analysis. A given test, or assay, may require, as an example, the use of an electrophoresis component, a thermalcycler component, a mass spectroscopy, and/or a gene sequencer. To further complicate matters, some assays require the manipulation of hundreds if not thousands of separate physical samples, each of which is desirably processed through a complex battery of tests, the collected data from which must then be analyzed by secondary analytical instrumentation before creating the intended result. In addition, the physical and logistical demands of the laboratory increasingly require the use of robotics as well as human participants in order to locate and deliver samples to the proper instrumentation.

The necessary complexity of the laboratory situation is worsened further by the difficult nature of communication. Each instrument, robot, and hardware or software application has input and output parameters/information which must be properly configured in order to be utilized. This may include experiment(al) parameters, commands, or catalog information. Unfortunately, because not all biological instruments were designed to communicate with each other, human intervention is frequently required to coordinate the activities of each instrument. Normally, a centralized information flow control system would be desirable in order to reduce human requirements and errors. However, because a particular laboratory environment may contain many different instruments, produced by different manufacturers and with different requirements, it is difficult to integrate instruments as they exist into a unified system. Furthermore, as new instruments are integrated into the laboratory environment with existing systems, additional difficulties may be encountered due to configuration limitations arising from a previous laboratory setup.

Summary of the Invention

The aforementioned needs are satisfied by the assembly of the present invention which, in one aspect, is comprised of a communications and control system for a biological laboratory having at least one user-interfaced client for controlling and monitoring biological assays and a plurality of instruments, each with at least one associated logical component, for obtaining identification information about biological samples.

The control system is comprised of a plurality of logical components for the instruments that receive commands from the laboratory management system and instruct the plurality of instruments to process at least one biological sample. The plurality of logical components includes at least one sample analyzer logical component for at least one sample analyzer that receives commands from the management system. The at least one sample analyzer is instructed by the at least one sample analyzer logical component to obtain identification information about the biological sample.

Another aspect of present invention is a central management component in communication with the laboratory management system, wherein the central management component identifies at least one pre-defined set of instrument instructions for controlling the operation of the plurality of instruments or for communicating with the plurality of instruments. This central management component has at least one data structure that identifies the logical location of the plurality of logical components including the at least one sample analyzer logical component. In one specific embodiment, additional instruments having associated logical components can be added to the management system by updating the data structure as to the logical location of the logical component of the additional instrument. The logical components can be added such that the management component is made aware of the logical location of the logical components associated with the instrument, and the additional instrument can be controlled by a user via the user-interfaced client that is connected to the central management component. This can be accomplished without requiring modification of the at least one pre-defined set of instructions or modification of the at least one user-interfaced client.

The invention also implements a control and communications system for a biological laboratory having at least one sample analyzer and biological data processor. The system comprises a client component having a user interface that allows a user to gain access and control or monitor the operation of biological instruments in the biological laboratory. It also involves a plurality of instrument components associated with each of the plurality of instruments.

The plurality of instrument components includes a sample analyzer that receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated sample analyzer. The purpose is to obtain identification information about the biological sample. An instrument component for biological data processor receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated biological data processor. The data processor analyzes the identification information obtained by the sample analyzer and provides signals indicative of the analysis.

The central management also includes at least one directory that provides information to the client component as to the logical location of the plurality of instrument components. The directory embodiment functions in such a way that the client component can determine how to access the plurality of instrument components by reference to the at least one directory. At least one directory includes information about one pre-set list of commands, at minimum, that the client component can use to send signals to the plurality of instrument components to induce the sample analyzer to obtain the identification information. The sample analyzer will then induce the biological data processor to analyze the identification information.

Another specific embodiment of the central management component is a messaging service that transmits messages to and from the plurality of instrument components and the client component. The messaging service is constructed in a standardized format wherein at least one directory and the client component are configured such that additional instruments can be added to the laboratory. Instruments are added to the laboratory by first associating an instrument component with the instrument itself. The system then updates at least one directory with information about the logical location of the instrument component and at least one pre-set list of commands for the instrument component. This can be accomplished without requiring the modification of the at least one pre-set list of commands or modification of the client component.

The present invention also presents a method of controlling the operation of a biological laboratory. The method involves associating an instrument component with a biological sample analyzer such that the instrument component translates received instructions to the biological sample analyzer. The biological sample analyzer can then be instructed to capture identification information about a biological sample.

Another specific aspect of the method is associating an instrument component with a biological data processor such that the instrument component translates received

instructions to the biological data processor. This function allows the biological data processor to be instructed to analyze identification information captured by the biological sample analyzer.

In addition, the method also maintains a directory of the information for each of the instrument components within the laboratory such that a user interface can determine by accessing the directory. The user can access the directory and determine which of at least one pre-set list of instructions are implemented by each instrument component, so that the user interface can access an instrument to implement a process.

The method will also add an additional instrument into the biological laboratory by associating an instrument component to the newly added instrument and updating the directory as to the instrument component. This allows the new instrument to be accessed by the user interface to implement a process without requiring modification of the user interface or to any of the at least one pre-set list of instructions. These and other objects of the present invention will become more fully apparent from the following description taken in conjunction with the accompanying drawings.

The aforementioned needs are satisfied by the assembly of the present invention which, in one aspect, is comprised of a control and communications system for a biological laboratory having at least one sample analyzer and biological data processor. The system comprises a client component having a user interface that allows a user to gain access and control or monitor the operation of biological instruments in the biological laboratory. It also involves a plurality of instrument components associated with each of the plurality of instruments.

The plurality of instrument components includes a sample analyzer that receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated sample analyzer to obtain identification information about the biological sample. An instrument component for biological data processor receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated biological data processor. The data processor analyzes the identification information obtained by the sample analyzer and provides signals indicative of the analysis.

The central management also includes at least one directory that provides information to the client component as to the logical location of the plurality of instrument components. The directory embodiment functions in such a way that the client component can determine how to access the plurality of instrument components by reference to the at least one directory. At least one directory includes information about one pre-set list of

commands, at minimum, that the client component can use to send signals to the plurality of instrument components to induce the sample analyzer to obtain the identification information. The sample analyzer will then induce the biological data processor to analyze the identification information.

Another specific embodiment of the central management component is a messaging service that transmits messages to and from the plurality of instrument components and the client component. The messaging service is constructed in a standardized format wherein at least one directory and the client component are configured such that additional instruments can be added to the laboratory. Instruments are added to the laboratory by first associating an instrument component with the instrument itself. The system can then update at least one directory with information about the logical location of the instrument component. This is done so that the client component can become aware of the additional instruments by accessing the at least one directory.

The invention also implements a communications and control system for a biological laboratory having at least one user-interfaced client for controlling and monitoring biological assays and a plurality of instruments, each with at least one associated logical component, for obtaining identification information about biological samples.

The control system is comprised of a plurality of logical components for the instruments that receive commands from the laboratory management system and instruct the plurality of instruments to process at least one biological sample. The plurality of logical components include at least one sample analyzer logical component for at least one sample analyzer that receives commands from the management system. The at least one sample analyzer is instructed by the at least one sample analyzer logical component to obtain identification information about the biological sample.

Another aspect of present invention is a central management component in communication with the laboratory management system, wherein the central management component identifies at least one pre-defined set of instrument instructions for controlling the operation of the plurality of instruments or for communicating with the plurality of instruments. This central management component has at least one data structure that identifies the logical location of the plurality of logical components including the at least one sample analyzer logical component. In one specific embodiment, additional instruments having associated logical components can be added to the management system by updating the data structure as to the logical location of the logical component of the additional instrument. The logical components can be added such that the

management component is made aware of the logical location of the logical components associated with the instrument, and the additional instrument can be controlled by a user via the user-interfaced client that is connected to the central management component. This can be accomplished by allowing the user-interfaced client to become aware of the additional instrument by accessing the data structure of the central management component.

The present invention also presents a method of controlling the operation of a biological laboratory. The method involves associating an instrument component with a biological sample analyzer such that the instrument component translates received instructions to the biological sample analyzer. The biological sample analyzer can then be instructed to capture identification information about a biological sample.

Another specific aspect of the method is associating an instrument component with a biological data processor such that the instrument component translates received instructions to the biological data processor. This allows the biological data processor to be instructed to analyze identification information captured by the biological sample analyzer. In addition, the method also maintains a directory of the logical location of each of the instrument components within the laboratory such that a user interface can determine, by accessing the directory, how to access an instrument to implement a process.

The method will also add an additional instrument into the biological laboratory by associating an instrument component to the newly added instrument and updating the directory as to the location of the instrument component. This allows the new instrument to be discovered and accessed by the user interface to implement a process by using the directory. These and other objects of the present invention will become more fully apparent from the following description taken in conjunction with the accompanying drawings.

The aforementioned needs are satisfied by various aspects of the present teachings. One aspect of the present teachings relates to a system for controlling the operation of a biological laboratory having a plurality of biological processing devices. The at least some of the biological processing devices have interrelated operational dependence. The system comprises a control system that monitors and controls the operation of the plurality of biological processing devices. The control system includes a standardized state logic having a plurality of standardized commands and an interrelationship database that correlates the interrelationship between a change of state of one of the biological processing devices to another of the biological processing devices. The control system, in response to receiving a state change from a first biological

processing device determines a desired state of a second biological processing device from the interrelationship database and sends a standardized state change command to the second biological device. The system further comprises a plurality of translation components associated with the plurality of biological processing devices. The plurality of translation components includes a customized instruction set that instructs the biological processing device to change from a first state to a second state in response to receiving the standardized state change command from the control system. The plurality of translation components include at least one sample analyzer translation component for at least one sample analyzer. The at least one sample analyzer translation component receives standardized commands from the control system such that the at least one sample analyzer is instructed by the at least one sample analyzer translation component to obtain identification information about the biological sample.

In certain embodiments, the standardized state diagram comprises a normal state and a fault state. The standardized normal and fault states for a given devices represent functionally similar device-specific normal and fault states. The normal state of the standardized state diagram comprises a task state and a standby state. The normal state of the standardized state diagram further comprises a completed state. The control system, upon receiving the standardized completed state from the first device, determines the second device's state in response to the first device's state. The control system instructs the second device to go into the task state if it is in the standby state.

In certain embodiments, the plurality of biological processing devices includes a sample storage device. In other embodiments, the plurality of biological processing devices includes a sample transfer device. The sample transfer device comprises a robotics device. In other embodiments, the plurality of biological processing devices includes a sample multiplexing device. The multiplexing device comprises a thermalcycler device. In other embodiments, the plurality of biological processing devices includes the at least one sample analyzer such as a mass spectrometer, a DNA sequencing device, an electrophoresis device, and an array device. In other embodiments, the plurality of biological processing devices includes a system monitoring device that monitors the operating conditions of the biological laboratory.

In certain embodiments, the translation component for a given biological processing device is functionally located in a front end component of the given device. In other embodiments, the translation component for a given biological processing device is functionally located in a translation repository accessible by the control system and the given biological processing device.

Another aspect of the present teachings relates to a method of controlling the operation of a plurality of biological processing devices adapted to facilitate biological assays. The method comprises monitoring the states of the plurality of biological processing devices in a standardized format. The standardized states of the plurality of biological processing devices are obtained from device-specific states of the plurality of biological processing devices by transforming the device-specific states into the standardized format. The method further comprises determining a course of action representative of an interrelated functioning of the plurality of biological processing devices based at least in part on the monitored standardized states of the plurality of biological processing devices. The method further comprises transmitting a set of standardized commands representative of the course of action to the plurality of biological processing devices. The standardized commands are transformed into formats recognizable by the plurality of biological processing devices.

In certain embodiments, monitoring the states of the plurality of biological processing devices comprises monitoring to see if the devices are in a standardized normal state or a standardized fault state. The normal state comprises a task state and a standby state. The normal state further comprises a completed state. Determining a course of action comprises receiving the completed state of a first device and in response, determining that a second device's state is to be changed based on the first device's completed state. The second device's state is to be changed to the task state if it is in the standby state.

In certain embodiments, transmitting the standardized commands to a given device comprises transforming the standardized commands to the device-specific commands at a front end component of the given device. In other embodiments, transmitting the standardized commands to a given device comprises transforming the standardized commands to the device-specific commands at a location that is not at given device.

Yet another aspect of the present teachings relates to a method of controlling the operation of a biological laboratory. The method comprises associating a translation component with an existing instrument such that the translation component translates transmitted state signals and received instructions based at least in part on the state signals so that the instrument can be instructed to facilitate capture of identification information about a biological sample being assayed. The method further comprises maintaining a directory of information for each of the translation components such that a user interface can determine by accessing the directory which instruments are available for use to determine an assay process based at least in part on the state signals and the

resulting instructions. the method further comprises updating the directory whenever an instrument is either added or removed functionally from the biological laboratory such that the user interface can adjust the manner in which the state signals and instructions are implemented in the assay process based on the availability of the instruments in the biological laboratory.

In certain embodiments, associating the translation component with the instrument comprises configuring a front end component of the instrument to perform the translation of the states and the instructions. In other embodiments, associating the translation component with the instrument comprises configuring an algorithm not on the instrument to perform the translation of the states and the instructions.

In certain embodiments, maintaining the directory comprises a listing of instruments available for use. The list of instruments comprises information about the instruments such as supported interface types, physical locations of the instruments, and identifiers that allow communication via a network. Updating the directory comprises updating the list of instruments.

Yet another aspect of the present teachings relates to a system for controlling the operation of a biological laboratory having a plurality of biological processing devices. The system comprises a control system that monitors and controls the operation of the plurality of biological processing devices. The control system includes a standardized state logic having a plurality of standardized commands and a database that correlates a change of state of a biological processing device to a corresponding command among the plurality of standardized commands. The system further comprises a plurality of translation components associated with the plurality of biological processing devices. The plurality of translation components includes a customized instruction set that instructs the biological processing device to change from a first state to a second state in response to receiving the standardized state change command from the control system. The plurality of translation components include at least one sample analyzer translation component for at least one sample analyzer. The at least one sample analyzer translation component receives standardized commands from the control system such that the at least one sample analyzer is instructed by the at least one sample analyzer translation component to obtain identification information about the biological sample.

These and other aspects of the present teachings will become more apparent from the following description taken in conjunction with the accompanying drawings.

One aspect of the present teachings relates to a system for providing a centralized user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples. The system comprises

an information component for instruments that includes information about the plurality of the biological processing instruments. The information for a given instrument comprises a list of the instrument's parameters that can be monitored and a list of the instrument's parameters that can be controlled. The system further comprises an interface application that generates the graphic user interface based on the information for one or more of the instruments. The interface thus generated is populated according to at least a portion of the instrument's monitor and control parameters thereby allowing the interface application to generate different interfaces based on a single adaptable template interface.

In certain embodiments, the centralized user interface for a given instrument allows a user to monitor and control the instrument. In one of such embodiment, the interface application uses the information about the plurality of instruments to generate an instrument management graphic user interface that lists instruments available for use.

In certain embodiments, the system further comprises an information component for samples that includes information about the plurality of biological samples. The samples' information component allows the interface application to generate adaptable user interfaces based on the samples' information. In one of such embodiments, the interface application uses the information about the plurality of biological samples to generate a sample management user interface that lists the samples in the biological laboratory. In one of such embodiments, the sample and instrument information are combined to allow the interface application to generate a study management user interface. Such user interface is adapted to allow a user to plan a manner in which data is taken during a run.

In certain embodiments, the system further comprises an information component for an analysis module that analyzes the data from one or more biological processing instruments. The analysis module's information component allows the interface application to generate adaptable graphic user interfaces based for viewing results of the analyzed data.

In certain embodiments, the plurality of biological processing instruments includes any combination of a sample storage device, a sample transferring robotics device, a thermalcycler device, a mass spectrometer, a sequencer device, an electrophoresis device, an array device, an analysis computing device, and a data storage device. In certain embodiments, the instrument component is part of an instrument directory.

Another aspect of the present teachings relates to a method for generating a centralized user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples. The method comprises receiving a request for a selected user interface; obtaining information

associated with the requested user interface; and generating the requested user interface by populating a template user interface based on the information obtained.

In certain implementations of the method, receiving the request comprises receiving a request for a user interface for one or more biological processing instruments. In one of such implementations, obtaining information comprises obtaining information about the one or more biological processing instruments from an instrument directory.

In certain implementations of the method, receiving the request comprises receiving a request for a user interface for one or more biological samples. In one of such implementations, obtaining information comprises obtaining information about the one or more biological samples from a sample list.

Brief Description of the Drawings

A computer-implemented biological data collection and analysis system will now be described with reference to the following drawings:

Figure 1 illustrates an example block diagram for an open system for controlling and monitoring biological data collection and analysis instruments over a network.

Figure 2 illustrates an example block diagram for an abstraction of the components of the system of Figure 1 along with common communications pathways.

Figures 3A and B illustrate example block diagrams describing two exemplary messaging processes.

Figure 4 illustrates an example block diagram describing components of the instrument directory of Figure 1.

Figure 5 illustrates an example block diagram describing components of an instrument software interface.

Figure 6 illustrates an example flowchart of a process that occurs when a new biological instrument is introduced into the system.

Figure 7 illustrates an example flowchart of a process that occurs as part of the process of Figure 5 when an instrument is connected to and registers with the system.

Figure 8 illustrates an example flowchart of a process that occurs as part of the process of Figure 7 when an instrument registers with the system.

Figure 9 illustrates an example flowchart of a process that occurs as part of the process of Figure 7 when a laboratory management system discovers instrument type information.

Figure 10 illustrates an example flowchart of a process that occurs when the system uses a connected and registered instrument to collect data results.

Figure 11 illustrates an example list of functions enumerated in a instrument service provider interface.

Figures 12A-B illustrate an example set of XML data describing container parameters for a biological instrument.

Figure 13 illustrates an example set of XML data describing container settings for a biological instrument.

Figure 14 illustrates an example set of XML data describing manual control parameters for a biological instrument.

Figure 15 illustrates an example XML schema describing the structure of data describing container parameters.

Figure 16 illustrates an example XML schema describing the structure of data describing container assay parameters.

Figure 17 illustrates and example XML schema describing the structure of data describing manual control parameters.

Figure 18 illustrates an integration of an exemplary application into an existing system in a manner similar to that for instruments.

Figures 19A and B illustrate possible methods of allocating resources during a biological assay process.

Figure 19C illustrates a possible embodiment of a hardware sharing layer that allows shared access to a hardware having a limited number of access ports.

Figure 20A illustrates a plurality of analytical devices being controlled by a control system via a translator.

Figure 20B illustrates a functional relationship between an instrument hardware component and an instrument service provider.

Figure 20C illustrates an exemplary standardized state diagram by which the control system operates to control the plurality of analytical devices.

Figure 21 illustrates one possible process that the control system can use to routinely monitor the states of the devices in a standardized format and issue standardized commands in response to those states.

Figure 22 illustrates one possible process that the control system can use to issue the standardized commands in response to the various standardized states.

Figure 23 illustrates one possible implementation of the exemplary standardized commands into device-specific commands.

Figure 24A illustrates a translator that translates the device-specific states into standardized states and the standardized commands into device-specific commands.

Figure 24B illustrates a translation process that translates a device-specific state into a corresponding standardized state.

Figure 24C illustrates a translation process that translates a standardized command into a corresponding device-specific command.

Figure 25A illustrates an exemplary system of analytical devices adapted to multiplex a DNA sample into a plurality of measurement devices, wherein each of the devices is controlled by a control system via a translator.

Figures 25B to 25J illustrate device-specific state diagrams showing various states associated with an exemplary feature of each device.

Figure 26A illustrates how a translation entity may reside at various locations to act as the translator between the device and the control system.

Figure 26B illustrates an exemplary translation for a sample storage device of Figures 25A and 25B.

Figures 27A and 27B illustrate how the control system utilizing the standardized format facilitates configuring and coordinating large number of devices.

Figure 28 illustrates an exemplary biological laboratory having a plurality of biological processing instruments that can be monitored and controlled by a user interface;

Figure 29 illustrates various functional features of the biological laboratory that can be represented by the user interface;

Figure 30A illustrates a graphic user interface (GUI) application that generates a GUI for a particular instrument based on information about that instrument, wherein the generated GUI is based on a common template GUI;

Figure 30B illustrates a web browser that could be used as a user interface;

Figure 30C illustrates a dynamic generation of an instrument-related GUI when a new instrument is added to the system;

Figure 31A illustrates an exemplary instrument information that can be used for generating the GUI for the exemplary instrument;

Figure 31B illustrates an exemplary GUI for monitoring the instrument of Figure 31A;

Figure 31C illustrates an exemplary GUI for controlling the instrument of Figure 31A;

Figure 32 illustrates an exemplary GUI for monitoring a list of samples;

Figures 33A-C illustrate exemplary GUIs that can be generated from instrument and/or sample information, thereby allowing representations of instrument and/or study management features;

Figures 34A-B illustrate exemplary GUIs that can be generated to display either raw data as they come out of instrument(s);

Figure 35 illustrates an exemplary GUI that can be generated to display an analyzed data;

Figure 36 illustrates a process that generates a GUI based on the available information;

Figure 37 illustrates a communication sequence and functionalities of a dynamic container creation service;

Figure 38 illustrates a sample snapshot from a data collection instrument;

Figure 39 illustrates a custom client that may use an exemplary function subscribeInstrumentEvents() to prepare itself to receive information from exemplary ICFStatusEvents() and ICFContainerStatusEvents() functions;

Figure 40 illustrates an exemplary flowchart that describes one possible implementation of a pulling and pushing process;

Figure 41 illustrates how a custom client can "push" container information to a service provider to persist in the service provider's database allowing it to be run at a later time;

Figures 42 - 46 illustrate various embodiments of XML schema for application and container definitions that may be used in implementation of service sets and container creation/utilization services; and

Figure 47 illustrates one method for implementing a web-browser based dynamically loaded user interface implemented using the JAVA language.

Detailed Description of Preferred Embodiments

Embodiments of the present invention are directed toward providing an open framework for biological instrumentation and analysis. As will be described below, embodiments of the present invention provide an open, extendable system for performing biological data collection and analysis. Further, embodiments of the system provide for various levels of abstraction of biological instrumentation, along with mechanisms for remotely discovering instrumentation requirements and results formats without requiring direct communication with individual instruments. Thus, wide varieties of instruments, including ones not yet created, can be introduced to the system without requiring lengthy setup time inefficient use of computing and human resources.

Referring to Figure 1, a block diagram of the open framework system is illustrated. A network 100 provides connection between the various components of the system. The composition of the network 100 can vary and can be, by way of example, the Internet, a

wide-area network, a local-area network, or a local Ethernet network. Unless otherwise indicated, the functions described herein are preferably performed by executable code and instructions running on one or more general-purpose computers. However, the present invention can also be implemented using special purpose computers, state machines, and/or hardwired electronic circuits. The term computer can include one or more co-located or geographically distributed computers.

In the illustrated embodiment, a principal investigator 102 inputs a series of data collection and analysis tasks, a "workflow," to a computer 102 running a laboratory instrument management system, or "LIMS." The principal investigator is generally the person with the primary control over the types of experiments performed in a lab and the direct research will take. Besides the LIMS computer, the principal investigator 102 utilizes a viewing computer 106, which provides reports on experiments and data collection. By utilizing the illustrated embodiment and thus being freed from having to control instruments individually, the principal investigator 102 is able to perform some, or all, laboratory experiments using only the LIMS computer 104 and the viewing computer 106.

A laboratory manager 108 is also illustrated, along with a second viewing computer 110. The laboratory manager has the principal task of monitoring the progress of data collection and making sure that the lab's instrumentation is in proper working order. While the principal investigator 102 utilizes the viewing computer primarily for the monitoring of data acquisition and analysis, the laboratory manager's viewing computer provides status reports on the various instruments connected to the system. By utilizing the system in this manner, the laboratory manager can monitor the state of an entire lab from a central location, making it less likely that instrument errors or maintenance needs will be missed.

The system also comprises software components which provide for instrument abstraction and instrument lookup. The first illustrated software component is the instrument directory 112, which maintains basic registration information about instruments connected to the system. The information maintained by the registry typically includes types of interfaces each instrument supports, instrument-specific information such as serial number and physical location, and identifiers and network locations of each instrument's service providers, as will be explained. In one embodiment, the instrument directory 112 also keeps track of the availability of instruments, deleting instrument records as instruments become inactive or go offline.

The instrument directory 112 also maintains information which can be used to locate instrument type servers on the network. The instrument type servers have the primary responsibility of maintaining and providing information about types of instruments

rather than instruments at the individual level. Instrument type servers are represented in Figure 1 as instrument type servers 114a and 114b, although other embodiments may include fewer or greater numbers of instrument type servers. The instrument type server provides the LIMS computer 104 information about instrument requirements, such as containers or assays that are supported by the instrument, so that the LIMS can prepare appropriate inputs for instruments. Additionally, instrument type servers provide parameters for manual control of instruments. The mechanisms of both the instrument directory and the instrument type server, as well as instrument abstraction and interfaces, will be described in greater detail below.

A messaging service 118 is also illustrated in Figure 1. As will be described in greater detail below, the messaging service provides the system a mechanism whereby control messages and results can be sent between the LIMS software, the instrument directory, any instrument type servers, and the various instruments without requiring direct communication between components. The messaging service provides a common interface whereby the components can asynchronously send messages between each other without sacrificing resources. Optionally, a data storage device 120 is included in the system for remote storage of results after data collection. Optionally, data is stored locally with the instruments. Another non-instrument component of the system are the instrument control computers 122. These computers contain service provider software 123 which provides control over the instruments as well as provide status updates to the system. As will be described below, service providers implement one or more pre-determined software interfaces which other system components can use to communicate with an instrument in a way that will be understood by each component. While each instrument typically has at least one service provider associated with it, in some embodiments an instrument is controlled or monitored by more than one service provider. The multiple service providers are allowed to run on separate machines. The control of instruments is described in greater detail below. Typically, most instrument communication is done via the instrument control computer, which relays this information to the instrument itself.

The system optionally comprises at least one instrument. Examples of these instruments are illustrated in Figure 1 in blocks 130, 134, 136, 140, 142, and 144. Some instruments, such as robotics apparatus 130, electrophoresis system 134, thermalcycler system 136, and a mass spectroscopy system 138, have associated biological samples 150 from which they gather data. Other instruments, such as the thermalcycler 144 or the robotic apparatus 130, optionally do not collect data; in the case of the thermal cycler 144, its purpose is to prepare biological samples for further testing by amplifying the material

present within the sample. Additionally, the system can optionally provide for instruments such as the analysis computer 142, which perform analysis without requiring contact with physical samples. Instead, the analysis computer mathematically 142 analyses already-collected data 160 to produce its results. In addition, pictured in blocks 132 and 140 are optional networking and service computers. These computers provide for the use of instruments that do not have their own network adapters or communications abilities, by creating a middleware interface between the instrument and the network. Typically, this is done when the instrument was designed to connect to its control computer through a serial interface rather than over a network. This instrument wrapping allows the system to interact with the networking and service computer as if it were the instrument and to not have to take into consideration peculiarities of the instrument's communications system. The instrument wrapping also allows service calls to be made to the instrument from outside the system, allowing a lab manager to check in instrument integrity remotely where otherwise the lab manager would have to perform checks physically at each instrument.

Figure 1 demonstrates some important and advantageous features of the system. Rather than requiring direct connections between instruments, the system described by Figure 1 utilizes a network, freeing a laboratory manager from requirements that instruments which communicate with each other must be in close proximity. Similarly, the use of the network allows monitoring laboratory technicians and investigators the freedom to perform assays and check instrument progress remotely. And because each component of the system is connected to the network more or less individually, and not in association with other components, instruments and computers are made easier to swap out and replace or upgrade. The distributed architecture of the system also helps prevent the failure of one component from interfering with the operation of other components.

Referring now to Figure 2, a block diagram of abstract system components is illustrated. Various illustrated components are represented as in communication with each other. As represented in Figure 1, one embodiment of these communications is through a common network using networking protocols such as TCP/IP. In another embodiment, the components may have dedicated connections between each other. The ordering of the exemplary laboratory components 210-230 is meant to represent basic important tools used by most biological laboratory manipulating genetic material in a simple order of use. Each is represented as a possible plurality of tools, and none should be take to represent any particular model or brand of component, but rather a broad class of components. The exact make up of a laboratory will likely differ from the one illustrated in Figure 2. However, Figure 2 demonstrates an important inventive aspect of the system

in that it abstracts and communicates with each type of instrument in a laboratory, providing control and monitoring throughout the experimentation process.

An exemplary ordering of abstract instruments is illustrated beginning with the sample storage 210. In a typical laboratory the component represented by block 210 would be, in one embodiment, a sample storage room, where biological samples are stored until being located and retrieved for use. The next illustrated component is the sample transfer 215. In one embodiment of a laboratory, this is a mechanized robotic apparatus which locates biological samples through the use of automated indicators, such as a bar code system. Next a sample multiplexer 220 is illustrated. In one embodiment this block represents a thermalcycler system, which allows biological samples to be amplified in number or quantity before being analyzed. Next illustrated is a sample analyzer 225, which can take the amplified physical sample from the component 220 and analyze it to produce digital data describing the sample. One example of this would be a gene sequencer; another would be a bio-informatics platform. Block 230 illustrates a sample data processor, which processes digital data taken from analysis machines and produces data describing secondary characteristics. And example of this type of component would be a mutation analyzer.

Associated with each of the components 210-230 is an illustrated service provider 235, which provides translation and communication for the instrument, in methods that will be illustrated later. These service providers are in communication, in turn, with the messaging service 205, which provides a general routing and clearing house for communications in the system. In one embodiment, the messaging service 205 is provided by a server implementing the Java Messaging Service® protocol. The use of the messaging service 205 provides easier expandability and quick connection to the system by allowing components to communicate without making direct connections between each other for most communications. Exemplary processes of sending messages through the messaging server are represented below in the discussion with respect to Figure 4.

The messaging service is also in communication with at least one client component 240. In one embodiment, a client component represents a laboratory management system, such as the LIMS computer 104 illustrated in Figure 1. The at least one client component 240 also represents viewers, such as viewing computers 104 and 110 in Figure 1. The at least one client component 240 is also in communication with the instrument directory 245, which stores and serves information about particular instruments; this corresponds to the instrument directory 112 in Figure 1. Direct communication with the instrument directory 245 is useful in one embodiment because the directory holds network addresses which are needed when sending messages through the

messaging service, as will be described below. More contents of the instrument directory will be described in greater detail below in the discussion with respect to Figure 4.

Also illustrated is at least one instrument type server 250, which corresponds to the instrument type servers 114a and 114b of Figure 1 and serve information about types of instruments to clients. These type servers are also in communication primarily through the messaging service 205. Not illustrated are certain limited communications between the instrument directory 245 and the at least one type server 250 and instrument service providers 240 for the purposes of registration. Similarly to the direct communication between clients and the instrument directory mentioned above, these limited communications are allowed so that the instrument directory can remain available independently of the messaging service. This is done so that communication can happen before addresses are known to the directory or messaging service, or while addresses are changing, which may cause messaging service confusion. The various communication processes between these components will be described in greater detail below.

While Figure 2 represents components of the system in an abstract way, it demonstrates a chief advantage of the open framework in that instrumentation is accessed and communicated with through consistent software pathways, e.g. the messaging service 205 communicating with the service providers 235. And because the system can support most types of instrumentation, as exemplified by components 210-230, the messaging diagram of Figure 2 is able to support the majority of control and monitoring work in any laboratory. Finally, the use of a messaging service means that components need to maintain very little information about each other or the networks on which they communicate, allowing simpler interoperability and addition of new components.

Referring now to Figure 3, a block diagram is illustrated demonstrating two exemplary messaging processes utilizing the message server. While the message passing in Figure 3 illustrates one embodiment of system, other embodiments may combine steps or add new ones. The Point-to-Point Messaging process illustrated in Figure 3a is commonly used when one member of the system, in this example a client, wants to send a message to another component, in this example an instrument. One particularly useful method of point-to-point messaging is the request/response method, whereby a request for an action is sent to a message receiver with the expectation that a response will be returned. The response can alternately contain new information, the status of the request task, or an acknowledgement message. In the example given in Figure 3a, a message is sent from a client in order to send a command request to an instrument; this is followed by a response sent back to the client. The use of requests

followed by responses is well known. In one embodiment, the client sending the request pauses any computation it is performing while waiting for the response; in another, the client continues with its processes while waiting for a response. In other embodiments, the message is not a command request, but rather information sent to the receiving component. In yet another embodiment, a response is not sent to the originating component.

The illustrated process begins with step 1, where the client software makes a request from the instrument directory to lookup the address of the instrument to which the client wants to send a command request. Continuing to step 2, the instrument directory returns an address to the client. In accordance with the abstraction principles at work in the system, the address given by the directory is that of the instrument's service provider, which is configured to accept messages from the messaging service, rather than the particular hardware of the instrument, which may have no knowledge of the larger system. In one embodiment this address is an IP network address. In other embodiments, the address is a system-defined identifier of the instrument, such as a set of one or more descriptive strings, that is later resolved into a network address by the messaging service, as will be understood. Having received the address, the client software then, in step 3, packages the message and address and sends these to the messaging service for forwarding. At this point, the client no longer has to concern itself with delivery of the message and can either wait for a response, or continue any other processing that it requires.

In step 4, the message is packaged and send to the instrument software. In step 5, the instrument software translates the received message into commands that will be understood by the instrument hardware and delivers these to the hardware itself. After processing the commands, the hardware may then give a response in step 6. In one embodiment, this is a data object representing the result of a data processing request sent in the client message. In another, this response is a result code announcing the success or failure of the received commands. In yet another, the response is a standard status report. Because the hardware may not know about the larger system, this response will usually be in a format defined by the hardware, and thus not necessarily recognizable by the client software. Thus, having received the response, the instrument software then, in step 7, does any translation necessary for understanding of the response and sends the response to the messaging service for return to the client. In step 8, the response is forwarded by the messaging service to the client, which, if it was holding computation while waiting for the response, resumes computation.

Referring now to the Publish/Subscribe Messaging diagram of 3b, a process is illustrated whereby a subscriber component, in this example a client, wishes to receive messages as they are made available by a publisher component, in this example an instrument. The publish/subscribe model is commonly used for status or event messages; in the illustrated example a subscription for status messages is used. This method allows the creator of the messages, the publisher, to inform more than one subscriber without having to send multiple messages or know the identity of any one subscriber. One example is a client wishing to know when a lengthy analysis process has been completed by an instrument. In this circumstance, the publish/subscribe method allows the instrument to signal that it has completed to any interested component without having to send a message to each component and while allowing new components to make themselves available for the completion message whenever it comes.

The process starts at step 1 where the instrument software lists topics to which it will be publishing messages in the instrument directory. In the illustrated example, this topic list includes a status topic. Typically, this is done during the instrument registration process, which will be described in greater detail below. As mentioned above, in the discussion with respect to Figure 2, there are occasional times when instrument software must communicate directly with the instrument directory; in one embodiment the registration process is one of those times. Next, at step 2 the client software requests a list of topics for the instrument from the instrument directory, and receives a list of those topics at step 3. In one embodiment, the topic list contains a general status topic and a general event topic; in another more detailed topics are listed. Next, in a step 4 the client, after deciding which topics it would like to subscribe to, sends a subscribe request to the messaging service. At this point, the client has completed a subscription and can now wait to receive messages from the messaging service.

At some later point, in step 5 the instrument hardware delivers a hardware status report to the service provider. The service provider does any necessary translation or addition of software status indicators and, having previously registered a status topic, then in step 6 publishes the status report to the messaging service under the previously-registered topic name. The messaging service, having previously received the subscription request from the client, then sends the status report to the client software at step 7, completing the process.

Figure 3 illustrates some advantage of the system over other systems. As mentioned before, the use of a messaging service allows components to communicate without requiring each component to maintain information about the components with which it sends messages. Additionally, the two messaging techniques illustrated provide

enough flexibility to allow direct component-to-component information sharing when necessary, while allowing for the more flexible publish/subscribe technique when direct communication is not necessary.

Referring now to Figure 4, a block diagram is illustrated describing an exemplary instrument directory 405. As mentioned above, the primary purpose of the instrument directory is to contain and serve information about instruments connected to the system to facilitate communication between system components. In the illustrated embodiment, the instrument directory is attached to a network 415, which can be of various networking types, as described above. The networking of the directory is particularly useful because it is in frequent communication with a number of system components in order that those components may communicate with each other. The illustrated embodiment relies upon a naming and directory interface for much of its communication. The use of a naming and directory interface allows system components to use system-specific names which can be translated by the interface into various types of network addressing schemes. Thus, as an example, system components can refer to each other through the use a standardized set of name strings, which the naming and directory interface can then translate into domain name server (DNS) or lightweight directory access protocol (LDAP) names without requiring understanding of these addresses on the part of the system components. In one embodiment the Java Naming and Directory Interface® (JNDI) is used to provide this function. Associated with the naming and directory interface 420 is a data storage 435 which provides a resource for storing various name/address bindings, as well as other information.

One important feature of the instrument directory is the instrument information registry 425, which features detailed information about individual instruments so that system clients may discover which instruments are connected to the system. Illustrated is information for an example instrument "A." In the illustrated embodiment, the instrument information registry maintains a physical location descriptor, here "aaaa." In a typical laboratory, this is done so that a system client can discover whether or not a particular instrument is physically located near a given set of samples and thus is able to process those samples. The inclusion of a physical location indicator increases the utility of the system by allowing it to monitor and control instruments located laboratories that are physically separated with less confusion. The information registry 425 also contains type information, here "λ" and group information, here "bbbb." In one embodiment, this information is used to identify the instrument as one of a general class of instruments and then to unique identify its type. One example would be an instrument of group "electrophoresis" with type "Applied Biosystems 3730." Through the use of these

descriptors, a client querying the system for instruments can gain information about every instrument of a particular type or group, filtering out only those instruments which are useful for a particular analysis. The type information is also useful for identifying which instrument type server is associated with the particular instrument, as will be described.

Next, the instrument information lists publishing information. As was described in the discussion with regard to Figure 3b, topic descriptors "cccc" and "dddd" are listed for a particular instrument; in one embodiment these are for events and status information. Also listed in the illustrated embodiment is a publisher name "eeee." The publisher name is included because, as discussed above with respect to Figure 1, some instruments may utilize more than service provider, and thus it is useful to keep track of which service provider will be publishing for a given instrument.

The final set of information is a list of registered service providers for the instrument. While an instrument can have only one service provider software, the system contemplates that multiple service providers may be used by an instrument, and that they may be executed on different computers at different network locations. Thus, as illustrated, the service provider a lists its network location as 1111. In one embodiment, this is a direct network location, such as an IP address. In another, the service provider location is a system-specific identifier which may be resolved into a network address using the naming and directory interface 420 or the messaging service.

In addition to providing a network address, the service provider information also lists available interfaces for use through that service provider. In one embodiment, these interfaces are pre-determined sets of functions which can be utilized by clients to control or monitor instruments. Because the interfaces are pre-determined and understood by clients of the system, they provide a method by which clients can send communicate with instruments while trusting that their communications will be understood. A few interface types of general interfaces are listed as examples, and not by way of limitation. One interface, the Framework Instrument Interface, is expected by the system to be implemented by at least one service provider for every instrument. This interface provides a common set of instrument control commands to perform activities such as: starting an analysis run, pausing a run, performing diagnostics, getting the instrument status, stopping a run, and shutting down. In one embodiment, the methods in the instrument interface are associated with a pre-determined instrument state model to which it is assumed every instrument service provider will comply. Other optional interfaces which can be implemented are illustrated, such as interfaces to allow access to service history details or instrument status, or to give an instrument information about containers the instrument will encounter in assays. The use of pre-determined interfaces, and their

listing on the instrument information registry, simplifies the process of implementing a service on an instrument, as an instrument software developer can know in advance what types of interaction a client will expect from each type of interface. Additionally, should a software developer decide to introduce a previously-unknown type of service at an instrument, new functions or interfaces can be added without the client software needing to understand implementations details.

Besides maintaining information about instrument and instrument service provider details, the instrument directory also provides a list 430 of active instruments. In one embodiment, this list exists so that instrument information can remain in the registry regardless of whether the instrument is available to perform analyses or be monitored. Thus, if an instrument is taken offline for some reason, the instrument can simply send a message to the instrument directory 405 requesting that it be taken off the active list. This will prevent clients from attempting to access the instrument until such a time as it is available again. One examples of a circumstances when this would be useful is the temporary disconnection of an instrument from the network for maintenance or software upgrade.

The instrument directory 405 also contains an instrument type server information registry 440, which contains a list of locations where instrument type servers can be found for each type of instrument. The type servers exist in order to serve clients information about controls available for each particular type of instrument, as well as any containers or assays that an instrument supports. In one embodiment, the types listed in the type server registry 440 correspond to types listed for each instrument in the instrument information registry 425, allowing a client to find an instrument type server for any instrument registered. Much like the service providers listed in the instrument information registry 425, the instrument type servers listed in the type registry 440 implement pre-defined interfaces. In one embodiment instrument type servers are implemented using the same interface objects as are used in service providers. While in one embodiment, separate type servers are used for each instrument type, in others instrument type servers are configured to deliver information about more than one instrument type. In yet another embodiment, a single instrument type server is implemented, the type server having information about every instrument type.

However, because these service providers exist specifically for the purpose of delivering type information, they typically have two pre-determined interfaces. One interface allows for the discovery by clients of supported containers and assays for a particular instrument type. In one embodiment this is done by returning XML data conforming to a pre-defined container and assay schema, upon request. The XML data in

this embodiment describes types of containers and assays supported by the instrument type. An example of this XML data will be described in greater detail below. Another interface allows for the discovery by clients of particular manual controls that the instrument type supports. Again, in one embodiment this interface, upon request, returns XML data conforming to a pre-determined schema, the data giving parameters that may be controlled by a client. An example of this XML data will also be described below. The use of type servers to provide type-specific information has a number of advantages over existing systems. First, the use of dedicated type servers reduces traffic to instrument service providers and save computation by the service providers. Secondly, the use of type servers allows clients to learn of updates to a particular type of instruments in one communication, rather than having to query each individual instrument. Finally, because container, assay, and manual control information are typically provided by a type server, a client can discover most of the information it needs to prepare experiments and controls for multiple instruments with very few communications.

Referring now to Figure 5, a block diagram of instrument hardware and software is described. An instrument service provider 505 is illustrated, comprising various components that are used in order to provide communication and control with the instrument hardware 510. In addition, networking and service middleware 515 is illustrated. This middleware conforms to the software running on the networking and service computers 132 and 140 as illustrated in Figure 1, and is used typically to provide networked communications where the instrument hardware was designed with only a serial connection interface. In the example, TCP/IP is used to communicate between the service provider 505 and the middleware 515, however as discussed above, other networking protocols can be used.

The instrument service provider 505 itself comprises multiple software modules. While the software modules are illustrated as separate and distinct, their functions may be combined into fewer modules or broken down into more modules while incorporating various aspects. In addition, the modules may represent differing degrees of execution independence, including separately-running applications, individual process threads, or dynamically-linked libraries being executed by another application. One illustrated software module is the request/response module 520. This module contains software calls and hooks in order to allow messaging to be sent and received from the messaging service. In one embodiment, this software module also controls publication of event and status messages. The framework instrument interface 525 corresponds to the interface discussed above with respect to Figure 4 which describes functions providing high-level control of the instrument. The service provider 505 also contains a state model 530,

which describes at an abstract level a plurality of execution states which any instrument in the framework is expected to conform to. In accordance with this state model, the service provider maintains a record of the current state of the instrument 535. In one embodiment, the functions enumerated in the Framework Instrument Interface correspond with states and transitions in the state model 530. The state model is described in greater detail hereinbelow in the section entitled: STANDARDIZED STATE INTERFACE.

Because the state model 530 is not intended to conform to particular commands of the instrument hardware, the service provider maintains a software module 540 which acts as an interface to the instrument hardware 510. In one embodiment, this module understands the various commands accepted by the instrument hardware 510. In another, the module 540 is able to accept information provided by the instrument hardware, such as status reports. In addition, the service provider maintains the current state of the hardware 545 in order to give contextually-appropriate commands to the instrument. Software module 550 provides translation between the state model and the hardware interface. In one embodiment, the translation module is a mapping between transitions in the state model 530 and pluralities of hardware commands. In another the translation module 550 translates information received from the instrument hardware and modifies it for use with the messaging service. The final illustrated module is the instrument directory interface 555, which provides communication with the instrument directory. A separate interface is provided for directory communication because, as mentioned above, certain direct communications are needed with the instrument directory rather than through the messaging service.

The service provider illustrated in Figure 5 demonstrates the advantages of the instrument wrapping and abstraction of the system. The numerous software modules of the service provider facilitate communication between clients and instrument hardware through the use of commonly-understood interfaces and messages. By allowing this communication to occur without requiring knowledge of the implementation of any particular instrument involved, the system once again invites simplified introduction and modification of instrumentation.

Figure 6 illustrates an example embodiment of a process of introducing a new biological instrument into the system. The process illustrates one advantage of the system in that any instrument can be integrated with the system regardless of whether it has been designed with the open framework in mind or not. Depending on circumstances, the process of Figure 6 may be performed by an investigator in a lab, a lab manager or technician, a programmer wishing to integrate the instrument into a laboratory using the illustrated system, or even the designer of the instrument itself. Starting at state

605, the new instrument is introduced into the laboratory. The ability of the instrument itself to recognize the open framework system are not important, and it is contemplated that instruments will vary. Some added instruments will be designed specifically with the open framework system in mind. Some instruments will pre-date the system or have control and monitoring systems designed without regard to being included in the illustrated framework. Other instruments will be duplicates of instruments already introduced to the laboratory. The term instrument will be used generally in the following discussions, so as to be inclusive of each of the various types of instruments mentioned above, as well as unmentioned instruments and biological instruments which are currently undeveloped.

Continuing to state 610, it is determined whether the instrument incorporates a computer. This is done in order to determine if the instrument could already support a computer corresponding to the instrument control computers 122 illustrated in Figure 1. The word "incorporate" should be understood to indicate that the instrument utilizes a computer for control and monitoring. Thus the term incorporate includes instruments with an on-board computer, instruments connected through physical serial connections to a computer, instruments connected via a network to an associated computer, and also instruments utilizing a plurality of computers. If the instrument does not incorporate a computer, the process moves to state 625, where a computer implementing interfaces to the open framework system is associated with the instrument. In one embodiment, this is done by providing a computer running an instrument service provider. After associating a computer with the instrument, the process moves to state 640, where a networking and service computer is added to the instrument, so that it can be attached to a network. In one embodiment, the networking and service computer is separate from the computer executing the service provider; in another, they are integrated. The process then proceeds to state 645, where the instrument is connected into the laboratory framework. The particular details of the process of state 645 are described below in the discussion with respect to Figure 7.

If, however, the instrument does incorporate a computer, the process continues to state 615, where it is determined whether or not the instrument supports the framework interfaces already. One method by which this could occur is if the instrument integrates a service provider. If the instrument does support the interface, the process proceeds to state 645, where the instrument is connected into the framework. If the instrument does not support the interfaces, the process continues to state 620, where it is determined whether or not the instrument interface can be modified to support the open framework. An example of an instrument that could support the interface would be an instrument which partially comprises a computer and for which a service provider has already been

written, either by developers of the instrument who wish to update it or third parties. If the instrument cannot be modified to support the interface, the process continues to state 625, where a computer is incorporated in order to support the interface. If however, the instrument's interface can be modified, the process continues to a state 630, where service providers are installed on the instrument computer, and then to state 645, where the instrument is connected to the laboratory framework. After connecting the instrument to the frame work, the process of Figure 6 ends. Figure 6 illustrates the advantageous flexibility of the framework in that instruments with differing degrees of computer integration or framework support can be incorporated into the system. The end result of any path through the process is an instrument that can be understood by the system through easily-understood ways.

Figure 7 illustrates an example embodiment of the process of connecting an instrument into the laboratory framework and having that instrument discovered by the LIMS. The process of Figure 7 corresponds to a process performed in state 635 of Figure 6. Because some states in the process in Figure 7 involve determinations by the LIMS in order to continue, delays of differing duration may occur before the continuation of the process at certain points. By including those waiting periods in the process description, Figure 7 illustrates the ability of the system to provide instruments which are ready to be discovered and used when the LIMS is ready for them, rather than requiring lengthy set up processes. The process starts at a state 705, where the instrument is connected to the network. Depending on the particular characteristics of the instrument, this may involve connecting the instrument itself to a network, or connecting its associated computer 122 with service provider to a network.

Next, the process continues to a state 708, where the instrument begins an installation procedure. In one embodiment, this can be done through a call to the instruments associated computer 122 and service provider 123. In another embodiment, the framework installation procedure occurs when the service provider software itself is installed on a control computer 122. In another embodiment the installation occurs when the service provider software is installed on the instrument itself. In yet another embodiment the installation procedure is performed by a separate installation application. In one embodiment, this installation application also creates the service providers for the instrument. The process continues to state 710, where the instrument registers its type and particular parameters with the open framework system. This process is described in greater detail in the discussion below with respect to Figure 8. The goal of registration is to provide relevant portions of the open framework with enough information that other instruments and analytical devices can design assays and workflows that make proper

use of the instrument without having to query the instrument itself or understand particulars about its implementation. In both these and subsequent states, in the case where the service provider is already up and running, the service provider itself typically does the majority of system communication, sending messages to the actual instrument only when necessary. Thus, in one embodiment, must of the registration that follows is done by the service provider.

After the instrument is registered on the system, the process waits at state 713 for the LIMS to perform an instrument query. Then the process proceeds to state 715 which is where the LIMS becomes aware that the instrument is in the directory. This can happen in a number of ways. The LIMS may query the directory to determine if any new instruments have been added. The LIMS also may query the directory based on other criteria, such as all instruments of a specific type, or all instruments in a specific physical location. In another embodiment, the LIMS may ask to receive updates when new instruments are added to the registry. In that embodiment, as instruments are registered, the instrument directory broadcasts a message informing the LIMS that there is a new instrument available for use.

Once the LIMS knows of the existence of the instrument, it may learn about the instrument to determine if it would be useful for a given workflow or assay. Until this time, the process waits at a state 717 for the LIMS to need information about the instrument. Continuing to state 720, the LIMS retrieves information from the directory relating to the network location of the service provider implementing the interface through which the LIMS wishes to communicate. Next, at state 725, the LIMS discovers the instrument's type, in order to create assays that correspond to the instrument. Further explanation of the type discovery is described below in the discussion with respect to Figure 9. After discovering the instrument type, the process of Figure 7 ends.

Figure 8 illustrates an example embodiment of the registration process, corresponding to the process of state 710 in Figure 3. Starting at state 805, an indicator of the location of the instrument directory is received by the installation software. In one embodiment, this indicator is the uniform resource locator of the instrument directory. In another embodiment the indicator is a local area network address. In yet another, embodiment, the indicator is an IP address. This indicator may be entered at different times; in one embodiment it is provided at the time the service provider software is installed, in another, it is provided at a later time, when a laboratory manager or other technician determines that the instrument should appear on the network. Next, at state 810, the installation software registers the name of the instrument, the instrument type, and any groups to which it belongs. The registered name may be any identifier, although

a unique name is desirable. The instrument type is an indicator, corresponding to a particular instrument type server, that indicates that the instrument supports all services associated with that type server. The group indicator is used to include the instrument in a general class of instruments; one example would be gene sequencers.

Next, at a state 820, the installation software loops through a registration process for each of the instrument's service providers. In state 825, it registers the name of the service provider with the instrument directory. In state 830, it registers indicators of the interfaces that service provider supports. While it is expected that one of the service providers for each instrument will support the Framework Instrument Interface, other interfaces are optional. The installation software then registers the network location of the service provider with the directory so that it can be found for later communication. Then, at state 840, the process repeats for each unregistered service provider.

Once each service provider is registered, the process continues to state 845, where the installation software registers topic names to which events and status messages will be published, as discussed above with respect to Figure 3. Next, the installation software, at state 850 registers container and assay parameters with the instrument type server for that instrument type. In one embodiment, if no type server exists for that instrument type, one is created and registered with the instrument directory. An example of a format of container and assay parameters is given below. Then, at state 855, manual control parameters are registered with the instrument type server.

APPLICATION AND INSTRUMENT DISCOVERY

Figure 9 illustrates an example embodiment of a process by which the LIMS discovers instrument type parameters. Because not all states in Figure 9 rely on each other for completion, other embodiments of the instrument parameter discovery can omit steps while incorporating inventive aspects. As illustrated in Figure 7, the process of Figure 9 can be performed by the LIMS upon registration. In addition, the process of Figure 9 can be performed at any time the LIMS needs type-specific information about an instrument. Starting at state 905, the LIMS locates the type server for an instrument. As described above, this can be done by querying the instrument directory to provide the address of an instrument type server associated with a specific type. Then, at state 910, the LIMS sends a message to the type server requesting data describing supported container types. Next, at state 915, the LIMS, having received a response containing XML data describing the container types, creates a container based on the received parameters. An example of a container detailed in XML is given in Figure 12. The process continues to state 920, where the LIMS sends a request to the type server for

supported assay parameters and then to state 925, where the LIMS prepares assay information based on the parameter information received in the type server's response. One example of XML assay data is given in Figure 13. The LIMS then, at state 930, requests manual control parameters from the instrument type server, and at state 935, receives XML data detailing the manual control parameters. One example of a set of manual control parameters is given in Figure 14. Figure 9 illustrates the ease of discovering a new instrument in the illustrated system. By reducing the process of receiving information about a instrument into three requests, the system reduces the amount of time needed for setup of an assay. Further, by utilizing XML data conforming to a specified schema for instrument parameters, the system is able to trust that a client will understand an instrument and be able to easily integrate it into assay workflows.

Figure 10 illustrates a high-level process that occurs as a client, such as the LIMS, utilizes the open framework to perform an assay. The process illustrated in Figure 10 demonstrates the ability of the open framework to automate and simplify the process of running assays in a biological laboratory. Beginning in state 1003, the LIMS becomes aware of a specific instrument connected to the framework system. As mentioned above with respect to state 715 of Figure 7, this can happen through querying the instrument directory or by receiving an indication that a new instrument has been added to the laboratory framework system. At a later point, in state 1005, the LIMS determines that it requires the use of this instrument. Then, at state 1010, the LIMS discovers parameters from the instrument's type server, such as container and assay parameters and manual control parameters. One example of this process is described above with respect to Figure 9.

Next, at state 1015, the LIMS determines if the instrument chosen at state 1005 requires data processed by another instrument. One example of this would be a software application for mutational analysis, which would require gene sequencer information in order to perform an analysis. In another embodiment, state 1015 would determine that the instrument required physical sample that must be processed in a specific way by another instrument, such as being processed by a thermal cycler. In yet another embodiment the LIMS determines at state 1015 that a physical sample is needed to be retrieved by a specific robot. If this is the case, then at a state 1020 the LIMS determines what data is needed. In one embodiment, this is done by inspecting the container data requested at state 1010. Once the data requirements are understood, at state 1025 the LIMS performs data (or sample) collected with the instrument identified at state 1015.

Whether another instrument was required at state 1015 or not, the process goes to state 1030 where the LIMS generates appropriate inputs for the instrument. In one

embodiment, this preparation is based on parameters discovered at state 1010. The process then continues to state 1035, where the LIMS, utilizing either the Framework Instrument Interface or parameters discovered at state 1010, initiates analysis by the instrument. In other embodiments state 1035 may conform to commanding non-analytical processes, such as for a thermal cycler or a robot. Finally, at state 1040, the LIMS receives data from the instrument. Figure 10 illustrates the simplicity of controlling and preparing an assay under the open framework. Because common interfaces are used, the process is relatively similar for any type of instrument, regardless of the analysis or task it performs. Additionally, the process of Figure 10 illustrates the ease of integrating the output of one instrument with the requirements of another, and in an automated way that can reduce human error.

Figure 11 illustrates exemplary functions implemented in one embodiment of the Framework Instrument Interface. While some embodiments utilize additional software interfaces for instrument control and monitoring, the Instrument Interface is provided here as an illustration of the types of functions that may be enumerated in an interface. The Framework Instrument Interface is one of the interfaces that a service provider may provide in order to allow for control and monitoring of an instrument at the abstract instrument state model level. As the other interfaces implemented by service providers, clients call functions in the interface through messaging requests to an instrument's service provider. In one embodiment, every instrument is expected to implement the functions enumerated by the interface.

Because the Interface is intimately associated with the state model, some functions illustrated in Figure 11 correspond to commands to transition between states in the instrument state model. One example of this is function 1155, StopImmediately(), which causes an instrument to immediately cease whatever analysis task it is currently performing and transition to from the Pauseable state into the Abort state. Other functions illustrated in figure 11 perform status tasks, such as function 1105, GetInstState(), which returns an instrument's state to the requesting client, or function 1110 GetInstStatus(), which returns a status response message to the requesting client. In one embodiment, function 1120, ManualControlCommand(), has a special role. It is utilized to pass a command code, usually in the form of a string, to the instrument in order to execute a manual control command. Thus, through function 1120, a client can issues commands in a more direct manner to the instrument, without being constrained by the state model. This is necessary for instrument-specific commands, and, as mentioned above, the parameters used to create a manual control command are received by a client by calls to the instrument's associated type server. The use of this particular interface is

advantageous in that it provides for a commonly-understood set of commands that work across all instruments. By tying the commands of the Framework Instrument Interface into the universal state model, a broad-based method of control is created that any client can trust in. This makes instrument addition to the system easier for clients, as they can trust that, at a minimum, certain functions will be supported by any new instrument.

Figure 12 illustrates an exemplary piece of XML data that describes parameters for a particular container and assay. The piece of data illustrated in Figure 12 is cut from a larger piece of data for ease of explanation; thus the example data is not necessarily representative as a complete container and assay data sample and is not necessarily in proper XML format. Additionally, only a few types of parameters are represented in the Figure as examples; this does not limit the number or type that can be used in an implementation of the system. While the use XML data to describe container, assay, and manual control parameters is advantageous because XML lends itself to extendibility, other data types may be used while incorporating inventive aspects. Typically, the parameter data illustrated in Figure 12 would be gained by a client by querying an instrument type server. This process of querying and receiving data would conform to steps 910-925 of Figure 9. Upon receipt of this data, a client would be able to create containers and assays for use in the queried type of instrument by interpreting the parameters listed in the data. In addition, the data also contains information which can be used to create a graphical user interface for the particular type of instrument. One the client had determined available parameters for the container and assays for that instrument, the client would then be able to create a set of XML data describing the particular containers and assays to be used.

Section 1210 of the data demonstrates descriptor parameters used to identify a container. The descriptor parameters can also be used in a graphical user interface used to set container information. Thus columns, such as "Container Name," "Container Barcode ID," and "Comment" are suggested for any user interface used to create containers. This is done in order that a lab technician creating a set of containers for testing will know which pieces of identifying information are necessary for proper tracking of the containers. Next, the data defines a number of entries that describe necessary container information parameters. Each parameter comprises a section similar to sections 1215, where descriptors for that parameter are listed. In section 1215, the container information is revealed to be the type of plate that is to be used in the instrument. Then, in section 1220, specific different values are given for this parameter; in this circumstance the client is given a choice between identifying a 96 well plate type of container or a 384 well plate type of container. If a graphical user interface is created

from this XML data, this section could be used to create a set of selection buttons or a drop down menu from which a lab technician could describe the type of plate that comprises the container.

Section 1225 demonstrates a different type of parameter. In this section, rather than define specific parameter values, the path of a Java class is given. Thus, in the illustrated embodiment, a client could create a software object instance of the Java class listed, and then use this class's methods to populate assay data for that container. Other embodiments may use different programming languages, as will be understood. This is advantageous over listing specific values as it allows complex data collection to be handled by software, simplifying the XML data, and allowing for the client to use the enumerated class to perform necessary calculations to create parameter data. In one embodiment, the class provides a user interface for entering data. In one embodiment, the class provides a method which creates XML data that can be sent to the instrument along with other container and assay parameters at the time an assay is to be performed. In yet another embodiment, a serialized version of the class itself is sent to an instrument's service provider, which is then utilized by the service provider to set instrument parameters.

Section 1230 is similar to section 1220, in that it also describes a list of parameters, this time describing a method of plate sealing for a container, and giving parameters "septa" and "heat seal" as choices. As above, these choices can be represented to a client in a user interface. Moving ahead to section 1235, a third type of parameter is shown. Here, the scheduling preference parameter is allowed to be any integer between a minimum and maximum value; in this case the range is 123-3210. Thus a client can see that a value must be set for this parameter within this range, and provide a user with the prompting to do so. In a graphical user interface, this might be done through the use of an input box or slider tool. Figure 12 illustrates just a few of the types of parameters that can be included in XML data for a instrument type. By representing parameters in this manner, rather than through less-expandable encoding schemes, the system allows for new types of parameter data to be easily added to the XML descriptions. In addition, by supporting dynamically-created software objects in parameter data, the system prevents the types of data being shared from being limited by any particular form of data.

Figure 13 illustrates an example of XML data describing a particular container. In one embodiment, this is created by a client after receipt of parameter data, such as in Figure 12, in order to instruct an instrument of the particulars of a container being used for an assay. The data can then be sent to an instrument or held by a client to be requested

when an instrument is ready for an assay. Figure 13 illustrates some XML code describing a container that follows the parameters of Figure 12. As in Figure 12, the data illustrated in Figure 13 is cut from a larger example, and is not meant to represent a complete container description, or proper XML formatting. Starting at section 1305, the container's descriptive information is given, including name, ID, owner, and comment. These descriptors correspond to the descriptor parameters given in section 1210 of Figure 12. Sections 1310 and 1315 describe data chosen based on parameters received in the XML data of Figure 12. In section 1310, which corresponds to the parameters described in section 1230 of Figure 12, the container data describes itself as utilizing a heat seal, and utilizes one of the values described in section 1230. In section 1315, a scheduling preference number is given, which is taken from the range described in section 1235 of Figure 12.

Section 1320 gives one example of a set of assay data. This data describes a particular well of the container (A1), gives a sample name and tracking ID, and describes the assay protocol used and how the results should be stored. Unlike the data described in sections 1305, 1310, and 1315, this data does not correspond to parameters explicitly given by any section of Figure 12. Instead, the data given was created by the use of an instantiation of the class described in section 1225 of Figure 12. Thus it is illustrated how a more complex piece of container or assay data may be created by a class described in parameter XML data.

Figure 14 illustrates XML data describing manual control parameters for an instrument, in this case an electrophoresis instrument. Data such as in Figure 14 will typically be obtained by a client after a manual commands request to an instrument type server. Then, as discussed above, command strings can be sent to an instrument using the Framework Instrument Interface. As in Figures 12 and 13, the data included in Figure 14 is cut from a larger example and is not meant to represent a complete set of manual control parameters, nor strict XML code.

At section 1405, and indication of a group of commands is given; in this example, the group is electrophoresis commands. In one embodiment, a given instrument may have more than one group of commands, such as commands describing an over used to heat a sample and a syringe used to deliver the same to and from a well. Sections 1410-1425 describe parts of the XML data for a particular command. In section 1410, the name of the command is given; in this case it is to set the power supply. In section 1415, a command string is given. This, along with the parameter given in section 1425, is what a client would send an instrument in order to manually turn the electrophoresis power on or off. Next, at section 1420, a comment describing the nature of the command is given.

The use of the name and comment indicators is advantageous if a client wishes to provide a user interface to the laboratory technician that is setting up the instrument. Then in section 1425, the parameters are given for the command. Here, they are a list consisting of on and off. In other embodiments, a list of different lengths or utilizing non-string values can be used.

Two other types of commands are illustrated in contrast to the one described above. In section 1445, a command is illustrated for an electrophoresis voltage setting. Rather than provide a list of discrete settings, the command parameters in section 1445 describe a floating-point value in the range 0.0-15.0. Thus, a client can submit a command request giving any value in this range and trust that the instrument will accept the command. In section 1460 a command is illustrated allowing a client to request a reading of the electrophoresis voltage from an instrument. As this is a request for a value, no parameters are listed and the client needs only be made aware of the command string.

As with the other XML data examples given, the example of Figure 14 demonstrates the flexibility of the system in that multiple types of instrument parameters can be described and delivered to a client, while giving enough information that a simple command can be sent back to the instrument to implement the client's needs.

Figures 15, 16 and 17 illustrate, in one embodiment of the invention, XML schemas defining the structure of XML data used to discover instrument parameters. Figure 15 illustrates a schema describing samples and container parameters. The schema illustrated in Figure 15 describes the structure of the XML data example illustrated in Figure 12. Figure 16 illustrates a schema for describing parameters for an assay and containers for that assay. The schema illustrated in Figure 16 describes the structure of the XML data example illustrated in Figure 13. Figure 17 illustrates a schema describing manual control parameters for an instrument type. The schema illustrated in Figure 17 describes the structure of the XML data example illustrated in Figure 14.

It will be appreciated that the concept of integrating an instrument into the laboratory system framework may be extended to include various software applications as well. Figure 18 illustrates an exemplary system where a new application 1520 is being added. The system may already have one or more applications 1500 integrated. Similar to the integration of instruments as described above, the applications 1500 may comprise application portions 1502 that interact with the system via interfaces 1504. In certain embodiments, the interaction with the rest of the system is performed via a network 1506.

Such integration and operation of the applications 1500 may be facilitated by an application directory 1512 that comprises information 1514 about the applications' 1500. Such an information structure may be similar to that of instruments described above. In

certain embodiments, the information about the applications may be stored in a same directory as that of the instruments. In certain embodiments, the information about the applications may be stored separately from that of the instruments. It will be appreciated that the information about the application and/or the instruments may be stored in one or more directories in any combination without departing from the spirit of the present teachings.

Figure 18 further illustrates that when the new application ("ZZ") 1520 is added to the system, information 1516 about the application 1520 is added to the directory 1512. Such addition of the information in the directory 1512 may be performed in a similar manner as that of the instruments described above. Once the application 1520 is integrated into the system and recognized by the system, it could be utilized in a variety of integrated manner to facilitated the biological assay process in a manner similar to that of the integrated instruments.

It will be appreciated that a biological laboratory configured in one or more of the foregoing configurations provide many advantages for processing biological samples and analyzing data gathered therefrom. Such advantages can be particularly appreciated when the number of instruments/applications and the samples are relatively large. In such a situation, a user attempting to coordinate a large number of biological assays may not utilize the resources of the laboratory in an efficient or convenient manner.

In one aspect, the integrated framework disclosed herein allows a user to plan an assay at a high-level of abstraction, and let the framework and its various systems handle and execute the various details. For example, the user may instruct the management system to perform a particular protocol on a particular sample. At this level of abstraction, the user may issue such instruction without having to be concerned with the details of how such protocol will be performed on the sample.

In one example of such an instruction involving high-level instrument protocols, a given sample is instructed to undergo a sample multiplying process followed by an electrophoresis measurement. This exemplary high-level instruction can then be implemented when the identified sample is retrieved from its storage location and moved to an available thermalcycler. The multiplication of the sample can then be carried out in the selected thermalcycler. The multiplied samples can then be processed by one or more selected electrophoresis instruments. Information that allow implementation of the high-level instruction may include details such as the location of the sample, the availability and condition of the selected thermalcycler and the electrophoresis instruments. As previously described, such details may be stored in the instrument and/or sample directory(ies).

In one aspect, such high-level abstraction concept includes a selection of instruments and/or software applications as needed. Such selection of the instruments and/or applications allow the laboratory system to utilize the resources in a more efficient manner by not tying up resources to a particular set of protocols. This concept of utilizing the instrument/application resources in an advantageously as-needed basis (as compared to the "committed resources" manner) is illustrated in Figures 19A-B.

Figure 19A illustrates a portion of one possible assay process 1530 that may be implemented in the aforementioned "committed" manner. The exemplary process 1530 is prescribed such that to perform exemplary processes A, B, C, and D, corresponding instruments A, B, C, and D are committed during the duration of the process 1530. Specifically, in step 1532, the process 1530 uses instrument A to perform process A. Upon completion of process A, the assay process 1530 proceeds to step 1534, where instrument B is utilized to perform process B. The exemplary process B may have two or more possible outcomes – a first outcome and other outcome(s).

As seen in Figure 19A, the assay process 1530 is instructed to utilize instrument C if process B's outcome is a first outcome. Otherwise, the assay process 1530 is instructed to utilize instrument D. Such a decision is made in a decision step 1536. If the answer is a "Yes," the outcome of B is a first outcome, and the assay process 1530 proceeds to step 1540 where instrument C is used to perform process C. If the answer is a "No," the outcome of B is the other outcome, and the assay process 1530 proceeds to step 1542 where instrument D is used to perform process D.

In certain embodiments, a similar assay may be performed in a different manner, wherein the resources are not pre-assigned as in the process 1530 of Figure 19A. The resources are instead assigned "just in time" to perform the desired functions of the assay. As shown in Figure 19B, an exemplary assay process 1550, instead of using a predetermined instrument in step 1552, uses an available instrument that is capable of performing process A. Upon completion of process A, the assay process 1550 proceeds to step 1554, where an available instrument capable of performing process B is used to perform process B.

As seen in Figure 19B, the assay process 1550 is instructed to perform process C if process B's outcome is a first outcome. Otherwise, the assay process 1550 is instructed to perform process D. Such a decision is made in a decision step 1556. If the answer is a "Yes," the outcome of B is a first outcome, and the assay process 1550 proceeds to step 1560 where an available capable instrument is used to perform process C. If the answer is a "No," the outcome of B is the other outcome, and the assay process

1550 proceeds to step 1562 where an available capable instrument is used to perform process D.

One can see that by committing an instrument "just in time" as need arises, the overall efficiency of resources utilization can be improved. It will be appreciated that such an assignment of resources may be facilitated and used more effectively when a laboratory includes a relatively large number instruments. It will also be appreciated that the depiction of the exemplary processes 1530 and 1550 in Figures 19A-B are in no way intended to limit the manner in which resources are utilized. That is, an assay process may utilize any combination of these two exemplary resource-allocation schemes without departing from the spirit of the present teachings.

In certain embodiments, the concepts of laboratory-universal recognition and support of the various instruments and applications, provided by the common architecture of the present teachings, are extended to include various files that store various information about samples being processed. Such files may include raw data files that are generated by the analytical instruments. The files may also include processed data files that result from processing of the raw data files by one or more applications.

Preferably, such sample-related files (and any other related files) are formatted to store arbitrary information such that a user accessing the files at a high-level does not need to worry about the details of file-specific details. Such file-specific details may be delegated to an entity analogous to the instrument directory described herein. Thus when the user wishes certain data or information to be stored, the implementation of such a storing process is carried out in a manner similar in spirit to the user-instrument interaction. Similarly, retrieval of data or information may be carried out in a similar manner.

Also preferably, the sample-related files are configured so as to be expandable to accommodate possible changes in the volume of information being stored. For example, a given instrument may be upgraded, and the resulting raw data may be of a greater resolution and encompass a greater dynamic range. Preferably, such changes in the raw data size are accommodated by the sample-related files. Such files that accommodate the changes in the exemplary data are also preferably backward-compatible, such that existing software applications can access the "changed" data file in a transparent manner. Such a functionality may be achieved, for example, by placing a header information for the data file, wherein the header includes the format of the data therein. Such an information may stored away from the data file itself in the form of an updated instrument information in the instrument directory.

In various embodiments, the various components of the LIMS system may be configured to permit access by multiple users. One means by which this may be accomplished is through the implementation of a hardware sharing layer that provides for simultaneous access. Thus for a selected instrument, multiple service tools and/or data collection software applications may access the resources of the instrument in a substantially transparent manner.

In one aspect, the hardware sharing layer may be used for the purposes of quality control, diagnostic, and checkout functions while simultaneously permitting user access or normal instrument runtime operations. Additionally, the hardware sharing layer may be used for the purposes of research and development on prototype instruments for which data collection software may not yet exist allowing a convenient means by which multiple users may access the resources of the instrument. An additional benefit of the hardware sharing layer is to provide a means to encapsulate knowledge of instrument-specific hardware interactions, allowing hardware developers to work to a single protocol for instrument communications.

In various embodiments, implementation of the hardware sharing layer (HSL) provides for convenient instrument accessibility and possesses the following characteristics: (a) The HSL allows simultaneous connection to a physical instrument by multiple clients. (B) The HSL may be used by Java and non-Java clients and further provide for web-browser based connectivity if desired. (c) The HSL may permit a selected client to be informed of the activity of other clients connected to a selected instrument or class of instruments. (d) The HSL may provide a deliverable communications implementation in the absence of the full data collection infrastructure. (e) The HSL is desirably configured to be substantially independent of underlying communications layers, instrument specific implementations, and third-party technologies to provide for improved compatibility and flexibility. Additionally, the hardware sharing layer may provide a means to configure resource limited communication paths such as firewire or serial connections to allow for shared resource access by implementation of a HSL front-end.

Figure 19C illustrates one embodiment of a hardware sharing layer architecture for a sequencer that may be used in connection with the system and methods of the present teachings. In one aspect, hardware and components may be viewed as a collection of "channels". For example, the various biological instruments such as gene sequencers may communicate through a multiplicity of TCP ports. Typically, these ports may include a command, data, and diagnostic port with each port having a channel through which messages may be addressed. Each message may comprise an envelope with a channel

address and a "payload communication" that may be instrument dependent. For sequencers, the payload may comprise a SCPI command. For an instrument which is controlled via a traditional DLL, the sharing layer may be responsible for translating the payload into the necessary DLL function calls.

In general, messages may be transferred between clients and the HSL using conventional TCP sockets or an equivalent communications protocol means. The use of TCP sockets is well understood and represents a nearly universal mechanism that is relatively easy to implement for many different programming languages and operating systems.

The system comprises an HSL server and one or more channel plug-ins which conform to a standard "Channel" interface understood by the server. In various embodiments, channel plug-ins may choose to implement some domain knowledge of the particular channels they manage, but the server itself need not necessarily maintain such knowledge. As an example, channels for the command and diagnostic ports on an exemplary sequencer may be aware that incoming data is newline delimited, and may use this knowledge in deciding when to bundle up a set of incoming bytes for relay to the client.

In various embodiments, messages transmitted through the HSL may be characterized by a "Type" attribute which defines the purpose of the message, and a "Channel" attribute which specifies the source or destination of the message. For messages that are intended to be transmitted to or from the HSL layer itself, such as a request for a channel enumeration, the Channel attribute may be designated as "HSL". Other channels may be instrument-specific. The content of the message is generally considered to be message-type dependent. The following table below outlines some exemplary message types, and is followed by sample XML-like message strings that may be used in implementation.

Message	Direction	Message 'Type' Attribute	Notes
Close Connection	To HSL	CloseConnection	Send just before closing client TCP port.
Subscribe To Channel	To HSL	SubscribeToChannel	Client will receive all output from this channel (see "Message" below)
Send Message To Channel	To HSL	ToChannel	Message content is channel-specific
Message From Channel	To Client	FromChannel	Channel has said something. All subscribers get the message.
Lock Channel	To HSL	LockChannel	Allow no other clients to send to this channel.
UnLock Channel	To Client	UnLockChannel	
Snoop Sends To Channel	To HSL	SnoopChannel	Listen to all Sends by other clients.
Snooped Message	To Client	SnoopedSend	Message sent from another client to the channel.
UnSnoop Sends To Channel	To Client	UnSnoopChannel	Quit snooping
Get Channel Lock State	To HSL	GetLockState	Does any client have a lock?
Channel Lock State	To Client	LockState	Response to the previous "get"
Command Failed	To Client	CommandFailure	Example: Attempt to send to a non-existent channel.
Get Channel Enumeration	To HSL	GetChannelEnumeration	
Channel Enumeration	To Client	ChannelEnumeration	
Get Connection Enumeration	To HSL	GetConnectionEnumeration	Get a list of all connections
Connection Enumeration	To Client	ConnectionEnumeration	

Message	Direction	Message 'Type' Attribute	Notes
Connection Opened	To Client	ConnectionOpened	All clients will be notified when any connection is opened or closed.
Connection Closed	To Client	ConnectionClosed	
Channel Locked	To Client	ChannelLocked	All clients will be notified when any locks are acquired or released.
Channel UnLocked	To Client	ChannelUnLocked	All clients will be notified when any locks are acquired or released.
Channel has gone off-line	To Client	ChannelOffline	e.g. TCP connection has been broken.
Channel has come on-line	To Client	ChannelOnline	e.g. TCP connection has been re-established.

Additional details of various messages:

SubscribeToChannel (Client -> HSL)

Output from the given channel may be routed to this client. Clients may subscribe to multiple channels on a single TCP port and then parse the "Channel" attribute on incoming messages, or may wish to open a different port for each channel.

Channel names may be fixed for each instrument type and are generally not expected to change. The GetChannelEnumeration (see below) message may be used to verify the channel names.

Example:

<HSLMsg Type="SubscribeToChannel" Channel="CmdPort">

</HSLMsg>

ToChannel (Client -> HSL), default ASCII format example

This message may be used to send information from the client to a channel. The default content format may be ASCII (bytes with numeric values between 0 and 127). In general, bytes between the <Content> </Content> tags will be routed to the channel. For example, a message composed in a text editor with the <Channel> attribute on a separate line from the first line of content will contain a leading newline (which is editor and operating-system dependent!) preceding any other content routed to the channel.

In the following example for a sequencer, note that there is a CRLF following the SCPI message content, since this may be required by the instrument.

Example:

<HSLMsg Type="ToChannel" Channel="DiagPort">
<Content>
DIAG:MST Off
</Content>
</HSLMsg>

FromChannel (HSL -> Client)

The following may be the response that would be sent by a sequencer diagnostic port to the previous example

Example:

</HSLMsg><HSLMsg Type="FromChannel" Channel="DiagPort">
<Content>OK DIAG:MST OFF
</Content></HSLMsg>

GetChannelEnumeration (Client -> HSL)

This message may ask the HSL to send a list of all Channels.

Example:

<HSLMsg Type="GetChannelEnumeration" Channel="HSL">
</HSLMsg>

ChannelEnumeration (HSL -> Client)

This message may be returned in response to the GetChannelEnumeration message. The individual channel names may be delimited by newlines. The response shown here is what may be expected for a sequencer.

Example:

```
<HSLMsg Type="ChannelEnumeration" Channel="HSL">
<Content>
DiagPort
DataPort
CmdPort
</Content></HSLMsg>
```

CommandFailure (HSL -> Client)

This message may indicate that the previous command sent by the client could not be parsed successfully, for example a tag or attribute was misspelled, or an invalid Channel name was provide.

Example:

```
<HSLMsg Type="CommandFailure" Channel="HSL">
<Content>Invalid channel</Content></HSLMsg>
```

ChannelOffline (HSL -> Client)

This message may be sent if a channel goes down. For a sequencer, this typically means the TCP connection has been lost.

```
<HSLMsg Type="ChannelOffline" Channel="DiagPort">
</HSLMsg>
```

ChannelOnline (HSL -> Client)

This message may be sent when a channel comes up. For a sequencer, this typically means the TCP connection has been re-established when the instrument has been powered-up.

```
<HSLMsg Type="ChannelOnline" Channel="DiagPort">
</HSLMsg>
```

It will be appreciated that the present teachings describe a system and methods for integrating laboratory instrumentation and applications to provide a unified control and coordination architecture under a common interface. The system may be adapted to a variety of different hardware and software components wherein the individual functionalities and input/output data types for each component are recognized and incorporated into a centralized control and monitoring system.

It will be further appreciated that the present teachings describe a system and methods for integrating laboratory instrumentation and applications to provide a unified control and coordination architecture under a common interface. The system provides mechanisms for detection of various hardware and software components wherein the individual functionalities and input/output data types for each component are automatically recognized and incorporated into a centralized control system that provides live monitoring of the operational status of available components.

STANDARDIZED STATE INTERFACE

One aspect of the present teachings relates to a control system that communicates with a plurality of devices and facilitates coordinated functioning of the devices based on the states of the devices. Figure 20A illustrates a functional block diagram showing a biological measurement related analytical system having a control system 2100 in communication with some possible types of devices associated with biological measurements. Such devices may include, by way example, one or more sample storage device 2102, one or more sample transfer device 2104, one or more sample multiplexing device 2106, one or more sample analyzing device 2110, and one or more data processing device 2112. The control system 2100 may also be in communication with one or more system monitoring device 2114. More specific examples of these devices are described below in the form of an exemplary sample analysis process.

It will be appreciated that any type of device not illustrated in Figure 20A may be included in the analytical system without departing from the spirit of the present teachings. Moreover, the presence of the type(s) of device as illustrated in Figure 20A does not necessarily mean that such a device is required in the analytical system.

In one aspect, the control system 2100 comprises a state monitoring system that performs a standardized state monitoring procedure for the devices it communicates with. It will be appreciated that by standardizing the state configuration of the devices, the control system 2100 can manage a relatively large number of devices in a simple manner.

As shown in Figure 20A, associated with each device is a translator that translates the communication between the control system 2100 and the corresponding device. Translators 2116, 2120, 2122, 2124, 2126, and 2128 correspond to the devices 2102, 2104, 2106, 2110, 2112, and 2114, respectively. In one aspect, the translator translates device-specific state signals from the device into standardized state signals. The translator also translates a generally reverse process wherein the standardized command signals, in response to the standardized state signals, are translated into device-specific command signals. Thus, one can see that the translators allow the control system 2100 to operate in a standardized mode, thereby allowing effective management of relatively large numbers of different types of devices.

In Figure 20A, the translators are depicted as being adjacent the devices It will be appreciated, however, that the translator may reside in any number of locations including, but not limited to, the front end component of the device, the control system, or some form of a repository accessible by the control system. It will also be appreciated that the translator (and the control system) may be implemented in various forms of a processor.

In general, it will be appreciated that the processors comprise, by way of example, computers, program logic, or other substrate configurations representing data and instructions, which operate as described herein. In other embodiments, the processors can comprise controller circuitry, processor circuitry, processors, general purpose single-chip or multi-chip microprocessors, digital signal processors, embedded microprocessors, microcontrollers and the like.

Furthermore, it will be appreciated that in one embodiment, the program logic may advantageously be implemented as one or more components. The components may advantageously be configured to execute on one or more processors. The components include, but are not limited to, software or hardware components, modules such as software modules, object-oriented software components, class components and task components, processes methods, functions, attributes, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

As previously described, Figures 1 to 5 illustrate a manner in which various devices can be integrated into a laboratory system. Such an advantageous manner is described in greater detail above.

Figure 1 illustrates an integration framework of a biological laboratory system having a plurality of exemplary devices indicated as a robot 130, an electrophoresis device 134, a PCR 136, a mass spectrometer 138, an analysis computer 142, and a thermalcycler 144. Such devices may be connected to a common network, and be

managed by a management system 104 via the common network. The management of the devices may be facilitated by an instrument directory 112 that includes information about the devices. Such information may include the types of interfaces supported by the devices, device-specific information such as serial number and physical location, and identifiers that allow communication via the network. In certain embodiments, the instrument directory 112 also includes a list of updated devices that are available for use.

The management of the devices may also be facilitated by an instrument-type service provider 114. In certain embodiments, the instrument-type service provider 114 provides the management system 104 with information about various devices' requirements so that the management system 104 can monitor and control the devices in an appropriate manner.

The management of the devices may also be facilitated by a messaging service 118. The messaging service 118 allows communication between the various management-related entities (e.g., management system, instrument directory, instrument-type service provider, and other entities described below), various devices, and other entities in the laboratory system. In certain embodiments, the messaging service 118 provides such communication without having to rely on a direct communication between components.

The management of the devices may also be facilitated by an instrument control component 122. In certain embodiments, the instrument control component 122 comprises a computing device configured to run a service provider software 123. Such software may be configured to provide control over the devices as well as provide devices' status updates to the management system 104. In certain embodiments, the service provider software 123 may implement one or more software interfaces that allow communication between the management-related components with the various devices in a meaningful manner.

The management of certain devices in the laboratory system may also be facilitated by networking/service computers (indicated as 132 and 140 in Figure 1). Such computers may provide improved communication with devices that do not have their own networking and communication capabilities. Thus, the networking/service computers emulate the desired networking and communication functionality for their respective devices.

Figure 2 illustrates one possible routing of communication between a client 235 and a plurality of exemplary devices (210, 215, 220, 225, and 230). The client 235 entity may include the management system described above in reference to Figure 1. In the exemplary communication scheme of Figure 2, an instrument directory 240 provides the

client 235 with information about the devices. Then, the communication between the client 235 and the device(s) is facilitated via a messaging service 205. The messaging service 205 may rely on an instrument-type service provider 245 to format the communication according to the devices' requirements.

Figures 3A and 3B illustrate two possible modes of communication between the various components. In a point-to-point method, a communication between a client and a device is performed in a generally chained manner. In an exemplary chain, a message from the client goes through a messaging service, through an instrument software, and then to the destination device (instrument hardware in Figure 3A).

Figure 3B illustrates a publish/subscribe messaging method. In an exemplary communication where the device wants to send a message to the client, the device sends a status signal to the instrument software that in response publishes that status (along with an appropriate topic) to the messaging service. The messaging service does not automatically relay the status message to the client. Instead, the client subscribes to the messaging service with an appropriate topic. The messaging service matches the topic from the instrument software with that from the client, and sends the corresponding status message to the client.

Figure 4 illustrates an exemplary instrument directory 405. Such a directory may include an instrument information 425 that contains information such as instrument name, physical location, type, group, publishing information, and other information useful for management of the instrument (device). The directory 405 may also include an instrument type provider information 440 that contains information about the instrument-type service provider described above in reference to Figures 1 and 2. The directory 405 may also include a list 430 of active instruments. The directory 405 may also include a storage device for storing information that facilitates various functions of the directory. The directory 405 is also shown to be connected to a network 415 so as to allow communication with other components.

Figure 20B illustrates a possible functional relationship between an instrument hardware component 509 and an instrument service provider 505 (described above in reference to Figure 1). The hardware is depicted to be in communication with a networking and service middleware 513. Such middleware may be an integral part of the instrument, or may be an emulating component (networking and service computer in Figure 1). The middleware 513 is in communication with other networked components via a network 508. The instrument service provider 505, also networked with the middleware 513, may include a plurality of functionalities, including a state model component 530 and

a hardware commands component 540. Various aspects of the state model component and the hardware command component are described below in greater detail.

From the brief description of the integration framework in reference to Figures 1 to 5, it will be appreciated that the present teachings may be implemented in such a framework in a number of ways. For example, entities described herein as "translators" may be incorporated as part of the instrument service provider (Figure 5). Similarly, entities described herein as "translations" may be part of the instrument interface (Figure 4). It should be understood, however, that the various aspects of the present teachings may be implemented in systems other than that of the aforementioned framework without departing from the spirit of the present teachings.

Figure 20C now illustrates one possible configuration of a standardized state 2200 having an external input 2216, and external output 2220, and an external exit 2222. The standardized state 2200 is depicted to be categorized into a NORMAL state 2202 and a FAULT state 2204. Within the NORMAL state 2202 are INITIALIZE state 2206, STANDBY state 2210, TASK state 2212, and COMPLETED state 2214. Throughout the description, the standardized states at the control system level are indicated by capital letters, and the device-specific states at the device level are indicated by lowercase letters. It should be understood that such convention is for the descriptive purpose, and in no way meant to limit the scope of the present teachings.

The standardized configuration of the standardized state 2200 may represent any one of the devices in communication with the control system 2100. Given such a set of standardized states, it is possible to construct and implement a standardized process for controlling the devices without worrying about device-specific parameters. Such device-specific parameters may include various states of the device as well as commands, understandable by the device, that change the device's states. It should be apparent that having such a standardized process facilitates easier arranging of functional relationships of the various devices.

It will be appreciated that the content of the standardized state 2200 described in reference to Figure 20C is exemplary, and is not intended to limit the scope of the present teachings. Various applications of systems of biological measurement devices may lead to such standardized state being configured in different manners. The concept of utilizing such a standardized state to effectively monitor and control a plurality of devices, however, is one aspect of the present teachings.

Figure 21 illustrates one possible process 2230 that utilizes the standardized state 2200. The process 2230 may be initiated (not shown) in the control system 2100, and the process 2230 may run in a continuous or periodic manner during the sample analysis

process. Although the process 2230 is depicted as a loop, it will be appreciated that such repetitive process execution may be implemented in any number of ways known in the art without departing from the spirit of the present teachings.

In Figure 21, the looping process 2230 comprises step 2232 where the process 2230 performs a standardized system monitor state check. In step 2234 that follows, the process 2230 performs a standardized state check for the devices recognized by the control system 2100. It will be appreciated that in certain implementations, the system monitor device (2114 in Figure 20A) may be treated as one of the analysis-related devices. The standardized state and command system and method described herein treats global devices (such as the system monitor device) as a compatible devices. Thus, it will be appreciated that steps 2232 and 2234 may be combined as a single step without departing from the spirit of the present teachings.

Following step 2234, the process 2230 issues standardized state commands in step 2236 based on the monitored states. In step 2238 that follows, the process 2230 waits for a predetermined time before looping back to step 2232. The predetermined time may be selected to be any value.

Figure 22 illustrates a process 2240 that may occur during a portion of a single cycle of the process 2230 of Figure 21. In particular, the process 2240 depicts one possible way the control system can issue standardized commands to a given device in response to the standardized states monitored. Thus, the process 2240 may occur during step 2236 of the process 2230 described above in reference to Figure 21.

The process 2240 begins at a start step 2242, and proceeds through a series of checks for the exemplary standardized states previously described in reference to Figure 3. The process 2240 in decision step 2244 determines if the device is in INITIALIZE state. If yes, then the process 2240 in step 2246 issues a command instructing the device to initialize. If no, then the process 2240 proceeds to act on the next standardized state for the device.

The process 2240 in decision step 2250 determines if the device is in STANDBY state. If yes, then the process 2240 in step 2252 issues a command instructing the device to remain in the standby mode. If no, then the process 2240 proceeds to act on the next standardized state for the device.

The process 2240 in decision step 2254 determines if the device is in TASK state. If yes, then the process 2240 in step 2256 issues a command instructing the device to perform its assigned task. If no, then the process 2240 proceeds to act on the next standardized state for the device.

The process 2240 in decision step 2260 determines if the device is in COMPLETED state. If yes, then the process 2240 in step 2262 issues a command instructing the next device (that functionally follows the COMPLETED device) to perform its task if able to. Thus, the process 2240 changes the standardized state of the next device to TASK. If the answer in the decision step 2260 is no, then the process 2240 proceeds to act on the next standardized state for the device.

The process 2240 in decision step 2264 determines if the device is in FAULT state. If yes, then the process 2240 in step 2266 issues a command instructing the device to attempt recovery or exit gracefully. If no, then the process 2240 in step 2270 sets the device's standardized state to FAULT since its state was indeterminate. The process 2240 ends at step 2272.

The process 2240 is an exemplary process that shows how the control system may issue standardized commands to devices based on the standardized states obtained from the same devices. It will be appreciated that such a process can be implemented in any number different ways utilizing different programming logic and algorithm.

The exemplary process 2240 of Figure 22 is performed for each of the devices in communication with the control system. Figure 23 now illustrates a summary table 2274 of the commands (in response to the standardized states) for the devices described above in reference to Figure 20A. It should be realized that the commands listed in table 2274 are exemplary for the purpose of describing a concept of how the control system can issue standardized commands to coordinate the sequence of different tasks performed by the devices.

The exemplary commands in response to INITIALIZE state are selected to be generally similar – initialize the device if not already initialized. The exemplary commands in response to STANDBY state are also selected to be similar – continue to standby. The exemplary commands in response to TASK state are also selected to be similar – continue the device's assigned task. It will be understood that different types of devices will likely have different means for initializing, standing by, and performing their respective tasks.

The exemplary commands in response to COMPLETED state allows transition of the sample between the devices in the analytical system. For the devices illustrated in Figure 20A, the sample analysis process progresses sequentially from the sample storage (2102) to the data processor 2112. Thus, if the sample storage is COMPLETED, the control system changes the next device (sample transfer) to TASK if in STANDBY state. Similarly, sample transfer device COMPLETED causes the control system to change the multiplexer to TASK if in STANDBY state. Similarly, multiplexer COMPLETED causes the

control system to change the sample analyzer to TASK if in STANDBY state. Similarly, sample analyzer COMPLETED causes the control system to change the data processor to TASK if in STANDBY state. If the data processor is the last device in the analytical sequence, its COMPLETED state may cause the control system to end the analysis process. For the system monitor device, the COMPLETED state may not be applicable since the device may be configured to monitor the analytical system during and possibly beyond the analysis process.

For the analytical devices, the exemplary commands in response to FAULT state are selected to be generally similar – attempt to recover or exit gracefully. Different devices may have different recovery and exit processes. For the system monitor device, the exemplary command in response to FAULT comprises a "system halt" to shut down the system. Some system monitoring devices and conditions that may give rise to such a command are described below.

As previously described, the control system monitors and coordinates the plurality of devices based on standardized states and commands via the translator. Figure 24A now illustrates a functional block diagram 2280 that shows one possible way of implementing such a functionality. A translator 2284 is functionally interposed between a control system 2282 and a device 2286. The translator 2284 interacts with the control system 2282 in standardized state (arrow 2292) and command (arrow 2290) formats. The translator 2284 also interacts with the device 2286 in device-specific state (arrow 2296) and command (2294) formats. As previously described such a translator may functionally reside in the device, control system, or anywhere that allows such translation between the control system and devices. One possible manner in which the translation can be implemented is described below by way of example.

Figures 24B-C now illustrate possible processes that may be implemented to carry out the function of the translator described above in reference to Figure 24A. Figure 24B illustrates a device-to-controller translation process 2600 that begins at a start step 2602. In step 2604 that follows, the process 2600 receives from the device a signal representative of a state of the device. In step 2606 that follows, the process 2600 determines a STANDARDIZED STATE corresponding to the device state. Such determination may be achieved by providing the translator with a predetermined assignment scheme that assigns the device's various states to their corresponding STANDARDIZED STATES. In step 2610 that follows, the process 2600 transmits the STANDARDIZED STATE to the controller. The process 2600 ends at a stop step 2612.

Figure 24C illustrates a controller-to device translation process 2620 that begins at a start step 2622. In step 2624 that follows, the process 2620 receives from the controller

a signal representative of a STANDARDIZED COMMAND intended for the device. In step 2626 that follows, the process 2620 determines a device-specific command corresponding to the STANDARDIZED COMMAND. Such determination may be achieved by providing the translator with a predetermined assignment scheme that assigns the controller's various STANDARDIZED COMMANDS to their corresponding device-specific commands. In step 2630 that follows, the process 2620 transmits the device-specific command to the device. The process 2620 ends at a stop step 2632.

Figures 25-26 illustrate an exemplary analysis process in terms of states and state-based commands facilitated by an exemplary translation. An exemplary analysis system 2300 comprises analysis related devices sample storage unit 2304, robotics 2306, thermalcycler 2310, mass spectrometer 312, sequencer 2314, electrophoresis unit 2316, array unit 2320, and data processor 2322. The analysis system 2300 further comprises a system monitor device 2302. The aforementioned devices are functionally connected to a control system 2308 through their corresponding translators (2326, 2330, 2332, 2334, 2336, 2340, 2342, 2344, and 2324, respectively).

It will be appreciated that the exemplary system 2300 may be a standalone system or a part of a larger system having a plurality of such exemplary systems. Such a larger system may operate the plurality of the exemplary systems independently from each other. In such operating configuration, the system monitor may monitor the operating condition(s) associated with more than one exemplary system. The larger system may also be configured such that components of the different exemplary systems can be "mix-and-matched" as needed. It will be appreciated that such flexibility in reconfiguring of components is facilitated by the standardized format of the states and commands that are processed by a control system.

Figures 25B-25J now illustrate some of the possible exemplary states that the various devices of Figure 25A can be in. The states are logically organized into groups that generally mirror the organization of the standardized state described above in reference to Figure 3. It will be appreciated that some of the devices may be relatively complex and can have many possible states other than those illustrated for the purpose of the description herein. Thus, a state diagram for each device depicts just few states representative of parameters that could be associated with the device. It will be appreciated that such representation of the device and its states is not intended to limit the scope of the present teachings.

Figure 25B illustrates an exemplary state 2350 for the sample storage unit (2304 in Figure 25A). Some of the possible states within the storage unit is depicted to be grouped into a normal state 2352 and a fault state 2360. Within the normal state 2352, states are

further grouped into a standby state 2354, a dispensing state 2356, and a completed state 2386.

A storage temperature parameter is selected, for the purpose of description, to represent possible states of the storage unit. Thus, the standby state 2354 may include an "idle" state 2370, a "nominal temperature" state 2372, and a "ready" state 2374. The dispensing state 2356 may include a "locate sample" state 2376, a "retrieve sample" state 2380, a "make sample accessible to robotics" state 2382, and an "end of dispensing" state 2384. The fault state 2360 may include a "non-nominal temperature" state 2390, an "attempt nominal temperature" state 2392, a "storage unit temperature fault" state 2396, a "dispensing error" state 2400, an "error recovery" state 2404, and a "dispensing fault" state 2406. The storage unit state 2350 may also include an entry point 2362, an normal exit point 2364, and a fault exit point 2366.

The storage unit may be configured such that in a typical normal operating sequence, the storage unit enters into the "idle" state 2370 via the entry point 2362. From the "idle" state 2370, the unit transitions to the "nominal temperature" state 2372 if the unit is able to maintain a nominal temperature. The unit then transitions to the "ready" state 2374, and remains there until instructed to go into the dispensing state 2356.

When the storage unit transitions from the standby state 2354 to the dispensing state 2356, the unit goes from the "ready" state 2374 to the "locate sample" state 2376. Subsequently, as the unit performs its dispensing function, it goes to the "retrieve sample" state 2380, followed by the "make sample accessible to robotics" state 2382 and the "end of dispensing" state 2384. The unit then transitions from the dispensing state 2356 to the completed state 2386, and from there exits via the exit point 2364.

During the standby and/or dispensing states 2354 and 2356, the unit may experience various types or malfunctions or errors. For example, during the "idle" 2370 to "nominal temperature" 2372 transition, if the unit is not able to maintain a nominal temperature, it may transition to the "non-nominal temperature" state 2390 of the fault state 2360 (instead of going to the "nominal temperature" state 2372). Once in the fault state 2360, the unit transitions to the "attempt nominal temperature" state 2392, and while there, attempts to establish and maintain a nominal temperature (as indicated by a loop 2394). If the attempt is successful, the unit transitions to the "nominal temperature" state 2372 to resume the standby operation. If the attempt fails, the unit transitions to the "storage temperature fault" state 396. From there, the unit exits the operational mode via the exit point 2366.

The unit may also experience an error during any states of the dispensing state 2356. Such an error can cause the unit to transition from the dispensing state 2356 to the

"dispensing error" state 2400 of the fault state 2360. Once in the "dispensing error" state 2400, the unit transitions to the "error recovery" state 2402, and while there, attempts to recover from the error (as indicated by a loop 2404). If the attempt is successful, the unit transitions back to the dispensing state 2356 to resume the dispensing operation. If the attempt fails, the unit transitions to the "dispensing fault" state 2406. From there, the unit exits the operational mode via the exit point 2366.

Figures 25C-J illustrate similar state transition diagrams for the other exemplary devices illustrated in Figure 25A. Figure 25C illustrates a state 2410 of the robotics device. The device state 2410 comprises a normal state 2412 having a standby state 2414 and a "transporting" state 2416, and a fault state 2420. For this device, a manipulator's function is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 8B.

Figure 25D illustrates a state 2430 of the thermalcycler device. The device state 2430 comprises a normal state 2432 having a standby state 434 and a "multiplexing" state 2436, and a fault state 2440. For this device, a lid position of the device is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 8E illustrates a state 2450 of the mass spectrometer device. The device state 2450 comprises a normal state 2452 having a standby state 2454 and a "measurement" state 2456, and a fault state 2460. For this device, a magnet's function is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 25F illustrates a state 2470 of the sequencer device. The device state 2470 comprises a normal state 2472 having a standby state 2474 and a "measurement" state 2476, and a fault state 2480. For this device, reagents-readiness is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 25G illustrates a state 2490 of the electrophoresis device. The device state 490 comprises a normal state 2492 having a standby state 2494 and a "measurement" state 2496, and a fault state 2500. For this device, an imaging system function is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 25H illustrates a state 2510 of the array device. The device state 2510 comprises a normal state 2512 having a standby state 2514 and a "measurement" state 2516, and a fault state 2520. For this device, an alignment of the array is selected to

represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 25I illustrates a state 2530 of the data processor device. The device state 2530 comprises a normal state 2532 having a standby state 2534 and a "data processing" state 2536, and a fault state 2540. For this device, a data acquisition system readiness is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

Figure 25J illustrates a state 2550 of the system monitor device. The device state 2550 comprises a normal state 2552 having a standby state 2554 and a "monitoring" state 2556, and a fault state 2560. For this device, a water-detecting flood sensor function is selected to represent the various exemplary states in a manner similar to that described above in reference to Figure 25B.

In one aspect, the various device-specific states and commands that cause transitions between such states are monitored and coordinated by the previously described standardized formats of the states and commands. In Figures 18 and 25A, each of the various devices has associated with it some form of a translator. Figures 26A-B now illustrate how such translators may be implemented in a manner that provides the exemplary translation functionality as described above in reference to Figures 24A-C.

Figure 26A illustrates a translation entity 2570 that may reside in a front end component 2572 of a device 2574 and/or in a translation repository 2576. It will be appreciated that the translation entity 2570 may be in any number of locations, functional or physical, associated with the analytical system. It may reside within a front end module (FEM) of a device, in a computer physically proximate one or more of the devices, in a computer that hosts the control system, in a computer physically removed from the analytical system (but functionally connected), or in any other possible devices, memory units, processes, and the like.

Figure 26B illustrates an exemplary translation 2580 that could be stored at any of the aforementioned locations and implemented to translate between a device and the control system. The translation 2580 could be implemented in any number of ways, including but not limited to, a lookup table, an algorithm, and the like. The storage unit is used as an exemplary device for the purpose of describing the translation 2580. Thus, the states illustrated in Figure 25B are referred to in the description of the translation 2580. It should be understood, however, that similar translation may be constructed for each of the devices in the analytical system. Furthermore, it will be appreciated that the logic of the exemplary translation 2580 is only an example. Any other logics (and their

implementations) that perform similar functions may be utilized without departing from the spirit of the present teachings.

The exemplary translation 2580 can be generally divided into two groups for the purpose of forming a functional link between the control system and the device. The first group may include the translation of device-specific states to the STANDARDIZED states. The second group may include the translation of the STANDARDIZED commands to device-specific commands.

In the device-specific to STANDARDIZED state translation group, exemplary translations may be constructed as follows. If the unit is in any of the "idle," "nominal temperature," "ready," or "standby" states, then the translator outputs the STANDBY state to the control system. If the unit is in any of the "locate sample," "retrieve sample," "make sample accessible to robotics," "end of dispensing," or "dispensing" states, then the translator outputs the TASK state to the control system. If the unit is in any of the "non-nominal temperature," "attempt nominal temperature," "storage temperature fault," "dispensing error," "error recovery," "dispensing fault," or "fault" states, then the translator outputs the FAULT state to the control system. If the unit is in the "completed" state, then the translator outputs the COMPLETED state to the control system.

In the STANDARDIZED to device-specific command translation group, exemplary commands may be constructed as follows. If the control system issues the STANDBY command, the device is instructed to go into its standby state (via the idle state) if not in the standby state. If the device is already in the standby state, then the device is instructed to continue its standby operation.

If the control system issues the TASK command, the device is instructed to go into its dispensing state if not in the dispensing state. If the device is already in the dispensing state, then the device is instructed to continue its dispensing operation.

If the control system issues the FAULT command, it could be because the storage unit is at fault, or because another device(s) is at fault. If the storage unit is already in the fault state, then it is instructed to continue the fault handling process as described above in reference to Figure 25B. If the unit is in the standby state, then it is instructed to remain in the standby state. If the unit is in the dispensing state, then it could be instructed to either to go into the standby state (while the other-device-caused fault is being worked on) if the fault affects the dispensing unit in some way, or to continue to dispense if the fault does not affect the dispensing unit. If the unit is in the completed state, then it could be instructed to ignore the FAULT command.

From the foregoing description, it should be apparent that by relegating the specifics of different devices to some form of translation entities and controlling the

plurality of devices in a standardized format, the control system's ability to coordinate the activities of the devices is greatly simplified. As is known, the number of devices in many analytical systems may be relatively large to provide high throughput of sample analysis. The standardized coordination by a control system is particularly valuable in such large systems.

Figures 27A-B illustrate an exemplary change in the configuration of a portion of an analytic system. Such configuration change may be facilitated by the simple standardized treatment of the devices. Figure 27A illustrates a first configuration 2600 that comprises a plurality of devices 2602 arranged in an exemplary manner as shown. A control system 2610 coordinates the operation of the configuration 2600 via a translation component 2604. The translation component 2604 comprises a list of devices in the configuration 2600. The device-specific translators that correspond to the list of devices may reside in any of the locations described above.

Figure 27B illustrates a second configuration 2620 that results from some change in the physical and/or functional aspects of the first configuration 2600. The exemplary changes are depicted as a removal of device labeled as "3" (indicated by box 2622), reassignment of device "4" as the input for device "16" (indicated by arrow 2626), and addition of a chain of devices "5," "19," and "29" (indicated by enclosed area 2624). The control system 2606 then adapts to the new configuration via a translation component 2630 that has an updated list of devices.

In certain embodiments, the added device may be recognized as a known device by the control system or some other process that integrates the various devices of the analytical system. In such situations, the device integration may be performed without a tedious human intervention, by updating the device list and retrieving a known translation from some library of translations. In other embodiments, the added device may not be recognized by the control system. In such situations, the device integration may require an additional step of constructing a translation between the new device and the control system. Once such translation is constructed, the device behaves as one of the known devices, and can be coordinated along with the other devices in the analytical system.

It will be appreciates that the present teachings describe an instrumentation interface that provides a standardized states of a plurality of devices to a control system. Coordinating the operation of the plurality of devices in an analytical system is simplified by allowing the control system to perform in terms of simplified standard states. Device specific states that vary for different types of devices are logically linked to the standardized states via a form of a translator. By having the control system not worry

about the specifics of a device, its ability to coordinate large number of devices is improved.

USER INTERFACE GENERATION AND IMPLEMENTATION

One aspect of the present teachings relates to a user interface system (UI) that provides means for a user to interact with a plurality of instruments and applications. The user interface system further facilitates monitoring and controlling of the instruments/applications in a centralized and a user-friendly graphical environment. Figure 28 illustrates a functional block diagram showing a system for biological analysis 3000 wherein an application programming interface (API) 3032 provides communication functionality between a user interface 3034 and several possible types of instruments 3002 associated with biological analysis. Such instruments may include, by way example, one or more sample storage devices 3010; one or more sample transfer robotics devices 3012; one or more sample multiplexing devices (e.g., a thermalcycler 3014); one or more sample analyzing devices such as a mass spectrometer 3016, a sequencer 3020, an electrophoresis device 3022, an array analysis device 3024; one or more analysis computing devices 3026; and one or more data storage devices 3030. In one aspect, the interface 3032 is configured to generate a user interface (UI) 3034 that facilitates the user's interaction with the exemplary instrument(s) described above. More specific examples of these instruments are described below in the form of examples.

It will be appreciated that any type of instruments not illustrated in Figure 28 may be included in the analytical system without departing from the spirit of the present teachings. Moreover, the presence of the type(s) of instruments as illustrated in Figure 28 does not necessarily mean that such a instrument is required in the analytical system.

In one aspect, the interface 3032 advantageously can be configured to provide a common graphical user interface (GUI) that dynamically adapts to the various parameters associated with the instruments. The interface 1032 may obtain the parameters for a given instrument in a manner described below. By delegating the details about the instruments to some database, the GUI formation may be greatly simplified since the user does not need to worry about the specifics of the instruments in configuring the application that generates the GUI.

In one aspect, dynamic generation of the user interface is accomplished in such a manner so as to simplify its creation by utilizing database-stored instrument and application definitions detailing input/output requirements for each type or class of instrument and application. This information is automatically associated with the appropriate field of the user interface thereby facilitating its generation and freeing the

user from having to know extensive details of the I/O requirements of each instrument/application while still allowing for relatively simple construction/configuration of the user interface.

Figure 28 also illustrates a sample 3004 being processed by the aforementioned exemplary instruments. The sample may be identified in any number of ways, including for example, a barcode or other unique identifier 3006. As described below in greater detail, a plurality of such samples may be managed via the interface 3032 in conjunction with the plurality of exemplary instruments.

Furthermore, Figure 28 also depicts the exemplary instruments having a physical portion and a programmatic portion. The programmatic portion may represent, for example, one or more processes that may be running in the instrument. The physical portion may represent the actual hardware portion of the instrument. As Figure 28 illustrates, the programmatic portions of the analysis-related instruments (exemplified as the thermalcycler 3014, mass spectrometer 3016, sequence 3020, electrophoresis 3022, and array 3024) may also be functionally coupled to the analysis computing device 3026 such that data can be passed therebetween and be processed remotely from the instrumentation. The computer 3026 may be functionally connected to the data storage devices 3030 so as to allow the computer 3026 to store and/or retrieve data therefrom. In certain embodiments, the data storage device 3030 may also be accessed by the instruments directly so as to store and/or retrieve data without requisite interaction by the computer 3026. The instruments and computing-related components arranged in the foregoing manners are representative of various configurations which allow biological measurements and analysis to be conducted using a centralized coordination system through which an investigator interacts using the user interface.

In general, it will be appreciated that the processors may comprise, by way of example, computers, program logic, or other substrate configurations representing data and instructions, which operate as described herein. In other embodiments, the processors can comprise controller circuitry, processor circuitry, processors, general purpose single-chip or multi-chip microprocessors, digital signal processors, embedded microprocessors, microcontrollers and the like.

Furthermore, it will be appreciated that in one embodiment, the program logic may advantageously be implemented as one or more components. The components may advantageously be configured to execute on one or more processors. The components include, but are not limited to, software or hardware components, modules such as software modules, object-oriented software components, class components and task components, processes methods, functions, attributes, procedures, subroutines,

segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

As described above, the instruments can be advantageously integrated and recognized via an instrument directory. As will be described below, such a directory having information about the instruments allows the interface (3032 in Figure 1) to advantageously provide an adaptable GUI for a given instrument.

Figure 29 illustrates how the user interface 3032 can be constructed to provide access to a variety of assay-related monitoring information and/or controls by including an instrument-interface capability, along with similarly common interfaces for samples, analysis, and other study-related features. The exemplary monitor and/or control features are depicted as being categorized as either relating to laboratory management functions 3042 or to study management functions 3044.

The laboratory management functions 3042 may comprise an instrument management function 3046 and a sample management function 3050. The instrument management function 3046 may comprise a monitor/control function 3060, a diagnostic function 3062, and a calibration function 3064. The sample management function 3050 may comprise an inventory function 3066 and a sample tracking function 3070.

The study management function 3044 may comprises a run planning function 3052, a data run function 3054, and an analysis function 3056. The run planning function 3052 may comprise a standard run function 3072 and a customized run function 3074. The data run function 3054 may comprise a monitoring function 3076 and an online analysis function 3080. The analysis function 3056 may comprise a detailed analysis function 3082 and a simulation function 3084.

It will be appreciated that the aforementioned exemplary functionalities of an assay may be represented to the user in the form of a GUI for informational and/or control purposes. By having the details of the underlying features such as instruments, samples, run planning, and analysis stored in accessible manner at some functional location, the application that generates the GUIs can be configured to generate a GUI that adapts and populates itself according to the specifics of a given feature. It will be appreciated that such a commonality of user interface improves the manner in which the biological laboratory can be managed in general.

In the description below, the common GUI adapting to a variety of underlying features is described in terms of instruments. It will be understood, however, that similar concepts may be implemented for managing samples and other assay-related features without departing from the spirit of the present teachings.

Figure 30A now illustrates a GUI application 3090 configured to generate a GUI. In one aspect of the present teachings, the application 3090 is configured to generate a common GUI that can be populated with particulars depending on the details of an instrument being interfaced with. Such common GUI (also referred to as a template GUI 3098) may be made accessible for the application 1090 to populate appropriately to generate the desired GUI.

Thus, the application 3090 generates a GUI 3092 for an instrument X 3104 based on available information 3100 for the instrument X. Similarly, the application 3090 generates a GUI 3094 for an instrument Y 3106 based on available information 3102 for the instrument Y. Such one-to-one relationship between a GUI and an instrument may be extended to any number of instruments.

As shown in Figure 30A, the information 3100, 3102 about the exemplary instruments X and Y 3104, 3106 may be functionally located in a database 3096 that includes instrument information 3096. Such instrument information database is described below in greater detail in the form of an example.

As further shown in Figure 30A, the application 3090 obtains the information 3100, 3102 for the exemplary instruments X and Y 3104, 3106 and generates the GUIs 3092, 3094 for X and Y, thereby allowing the user to monitor and/or control the instruments X and Y 3104, 3106. In certain embodiments, the interaction between the GUIs and the instruments may be performed via the application 3090. Alternatively, the GUIs may be configured to bypass the application and interact directly with the instruments. In either of these two alternatives, the interaction may be either directly or via a controller 3108.

Figure 30B illustrates that in certain embodiments, a user interface application 3214 may be configured to launch user interfaces 3222. Such interfaces may comprise a GUI 3224, a web browser based user interface 3226, or any combination thereof. The use of web browsers as a basis for the user interface may be advantageous in certain platforms that support the user interface application 3214. In certain embodiments, such web browsers may be implemented using the HTML code known in the art.

The user interface application 3214 may populate the GUI 3224 and/or the web browser 3226 using information obtained from an information source such as an instrument directory 3210 in a manner similar to that described above in reference to Figure 30A. The directory 3210 is depicted to include available information 3212 for a plurality of exemplary instruments 3216 "X1" "X2," etc. Also similar to the configuration of Figure 30A, the interaction between the instruments 3216 and the user interface application 3214 may be either directly or via a controller 3220.

Figure 30C illustrates a system 3240 of certain embodiments configured to dynamically generate a GUI 3254 when an exemplary instrument (denoted as "YY") 3242 is added to the system 3240. A directory 3246 may be updated with information 3250 about the instrument "YY" 3242 in a manner described above. Such updating of the directory 3246 may be facilitated by a management system 3244.

The addition of the instrument "YY" 3242 to the system 3240 may cause the management system 3244 to discover the instrument "YY" 3242 in a variety of ways. For example, the management system 3244 may routinely check the directory 3246 and note the additional entry. In another example, the addition of the instrument's information 3250 in the directory may be accompanied by a broadcast to the system 3240, informing the system 3240 of the addition.

The management system 3244 may be configured such that upon such detection of an added instrument, it may invoke a GUI application 3252 to launch a GUI 3254. The GUI 3254 generated in such a manner may include the instrument's available protocol as well as other features that facilitate the sample assay process. The GUI application 3252 may be configured to generate the GUI 3254 by populating a common template GUI, based on information 3250 for the instrument "YY" 3242, in a manner similar to that described above in reference to Figure 30A. Thus, one can see that such a functionality allows a user to add a compatible instrument to the system 3240 and be provided with a visual interface that aids in incorporating the added instrument into an assay process.

Figures 31A-C illustrate an exemplary instrument information 3112 and its corresponding GUIs 3114 and 3120 that may be generated by the application 1090 described above. The exemplary instrument information 3112 may be part of an instrument directory 3110. The instrument directory is described above in reference to the system framework.

As shown in the instrument information 3112, the exemplary information may comprise an instrument identifier. In the exemplary information, the instrument is designated as "A" that represents an array device.. The information further comprises available monitor parameters and available control parameters for the array device.

The exemplary available monitor device may comprise the parameters listed in Figure 31A. As shown in Figure 31B, such parameters populate the GUI 3114 by the application 3090. In certain embodiments, the order of parameters in the information 3112 may determine the manner in which the they are arranged on the GUI 3114. Additional sub-parameters associated with the parameters may also determine how the parameters are displayed on the GUI 3114. For example, exemplary parameters numbered as 15 to 20 have associated with them a series of input values to be interpreted

in the following exemplary manner: first sub-parameter is the unit; second and third sub-parameters are the lower and upper limits for actual value (fourth sub-parameter) and set/nominal value (fifth sub-parameter). Thus, for the exemplary parameter "EP Voltage," the unit to be displayed is "KV," and the lower and upper limits of a scale to be displayed are 0 and 15 KV, respectively. The fourth sub-parameter of "EP_V_actual" is a value of the actual voltage of instrument "A" 3116 (1.0 KV in the exemplary GUI of Figure 31B). The fifth sub-parameter of "EP_V_set" is a value of the set voltage for the instrument 3116 (3.0 KV in the GUI of Figure 31B).

The ranges of parameter values in the foregoing example may be formatted so as to instruct the application to display them in a certain manner. It will be appreciated that there are many number of ways information can be displayed on a GUI. It will be further appreciated that the information for instruments may be formatted in any number of ways to provide any number of display formats of the GUIs without departing from the spirit of the present teachings.

Figure 31C illustrates an exemplary GUI 3120 launched by the application 3090 based on the information 3112. The exemplary available control parameters may comprise a laser operation and a comment function. The laser function may be formatted in the following exemplary manner: first sub-parameter is the unit; second and third sub-parameters are the lower and upper limits to be displayed proximate a fourth sub-parameter of a user input value. The laser function may comprise one of the following exemplary choices: "Set Power"; "Apply Power"; and "Shut Down." As previously described, in certain embodiments, the manner in which the control parameters are displayed may be determined by the format of the listed control parameters. In the exemplary GUI 3120 described above, the one or more control commands selected through the GUI may be sent to the exemplary instrument 3116.

Figure 32 now illustrates an exemplary GUI 3130 launched by the application 1090 based on a sample list 3132. The sample list 3132 could be considered to be a similar "information" about the samples analogous to the instrument information described above. The sample list 3132 could contain information about the inventory of substantially all the samples within the system, or about the whereabouts of a particular sample. The sample list 3132 could be facilitated by the sample-identifying means such as the barcode 3006 described above in reference to Figure 28.

Figures 33A-C now illustrates some exemplary GUIs that may be launched by the application 3090 based on some combination of instrument information and sample information. Such combination of information can facilitate the various laboratory and study management functions described above in reference to Figure 29.

Figure 33A illustrates an exemplary GUI 3140 that may be launched by the application 3090 based on the collection of instruments' information (1096 in Figure 30A). In particular, the exemplary GUI 3140 is configured to list instruments that may be used with a particular type of an exemplary activity ("Aliquot DNA") with a particular type of an exemplary assay process ("ProcessRun02"). As shown in Figure 33A, the GUI 3140 can also list instruments associated with other exemplary activities "Sample Setup" and "Perform PCR," with their respective exemplary processes "ProcessRun01" and "ProcessRun03."

Figure 33B illustrates an exemplary GUI 3142 that may be launched by the application 3090 based on information provided by an activity manager 3144. The activity manager 3144 may be a process or a database that combines information from the sample list 3132 and the instrument information 3096. The exemplary GUI 3142 lists the samples ("C1234567890") that are to be processed for the exemplary activity "Aliquot DNA" via the exemplary process "ProcessRun02."

Figure 33C illustrates an exemplary GUI 3146 that may be launched by the application 3090 based on information provided by a run scheduler 3150. The run scheduler 3150 may be a process or a database that combines information from the sample list 3132 and the instrument information 3096 to schedule processing of the samples via the available instruments for one or more types of data-taking runs. As shown in Figure 33C, the particular GUI 3146 lists runs to be performed by an exemplary instrument "3100test1."

It will be appreciated that the various exemplary GUIs described above in reference to Figures 31-33 can be considered to be various populated form of a single GUI launched by the application 3090. The various populations are based on the instrument and/or sample information stored in a database separate from the application that provides the interface with the user. The information provided to the application may be processed by some intermediate process or database such as the exemplary activity manager (3144 in Figure 33B) and the run scheduler (3150 in Figure 33C). In either of these two cases, however, the information provided to the GUI application 3090 is generally stored separately from the application 3090, and can be considered to be another form of database on which the GUIs are based upon.

This concept of providing the GUI application with information for the adaptive populating GUI can be extended to data presentation. Figure 34A illustrates an exemplary GUI 3160 displaying a raw data from an instrument "my3100." The exemplary GUI 3160 may be launched by the application 3090 based on the instrument information 3096 that

corresponds to an instrument 3162. In particular, the instrument 3162 comprises the instrument "my3100."

The raw data displaying GUI 3160 displays the data as it is being output from the instrument 3162. As previously described, the manner in which the GUI display a given parameter could be determined by the format of the parameter in the instrument's information. Thus, the exemplary raw data parameter could be formatted so as to make the application 3096 interpret the raw data output from the instrument 3162 as a streaming graph-format display.

Figure 34B illustrates another exemplary data-displaying GUI 3170 that may be launched by the application 3090 based on the data from an exemplary instrument "kaitwo" 3172. Information about such an instrument may be obtained from the instrument information 3096. The information about the instrument 3172 may include an output data format that allows the GUI 3170 to display the data in a two-dimensional "Gel display" format.

Figure 35 illustrates another exemplary data-displaying GUI 3180 that may be launched by the application 3090 based on the data output from an analysis module 3182. In certain embodiments, the analysis modules that process the data from the instruments could be treated like the instruments in terms of integration and being listed in some form of a directory. That is, the analysis modules could be listed in a database similar to the instrument directory described above. Such a database could also list each analysis module's available outputs.

Thus, the exemplary GUI 3180 displays a post-instrument(s) processed data from the analysis module 3182. The analysis module 3182 can also output the processed data to a result database 3184 for later GUI display and/or other form or presentation.

The various exemplary GUIs described above can be generalized as being a single or a few GUIs that can be populated by information obtained from some source accessible by the application that generates the GUI. In certain embodiments, the application generates a single GUI that adapts to the various information.

Figure 36 now illustrates a process 3190 for generating the aforementioned GUIs. The process 3190 may be performed by the GUI application 3090. The process 3190 begins as a start step 3192, and in step 3194 that follows, the process 3190 receives a request for a user interface. Such a request may originate from an existing GUI. For example, a user may request additional information about the instrument management (3046 in Figure 29) from an existing GUI that displays the lab management summary 3042. Similarly, the user may request an additional information on a particular instrument

from the thus-created GUI (of instrument management 3046), thereby creating the monitor/control functionality 3060 GUI.

In step 3196 that follows, the process 3190 obtains information associated with the user interface. In step 3200 that follows, the process 3190 generates a GUI populated with the information obtained. The process 3190 ends at a stop step 3202.

Control And Management Services

In one aspect, the present teachings define a set of services (e.g. instrument and applications services) that are commonly supported within the LIMS system. These services desirably aid in control and management simplification across different instrument and application platforms and facilitate integration of these components into the unified framework of the LIMS system.

In various embodiments, the service sets provide means for control and monitoring of various components, including instruments and applications, within the LIMS environment. Additionally, these service sets may be desirably configured to enable remote connectivity via a network or other communication means. In general, components of the LIMS system may desirably implement a standardized instrument or application service interface thereby providing a consistent set of interfaces which facilitate component integration into the LIMS system. In instances where an interface may not be applicable to a certain component, instrument, or application type, the component may recognize and indicate that the interface is not-applicable.

In one aspect, the design and implementation of the component application programming interface(s) (API) provide a framework of commonality in which a user interface (GUI) may be implemented on the basis of an instrument services contract. Implementation in this manner may be substantially independent of any instrument specifics while still being able to control and monitor a selected component configured to utilize the service set.

The following description details various aspects of the service sets and provides examples of how these services may be implemented in the LIMS system. These functionalities are meant to illustrate possible configurations which may be utilized by the LIMS system and should not be construed to limit its functionality.

In one aspect, a service set may comprise a dynamic container creation service. Figure 37 illustrates the communication sequence and functionalities of the dynamic container creation service 4000. In one aspect the dynamic container creation service 4000 functions to provide application dependent container type information. The type of

information that is gathered for each container typically depends on analysis application or instrument. In one exemplary application, when there was a new type of analysis application is made available which is to be desirably integrated into the system, other applications, such as data collection software, may need to be adjusted or modified to support the new analysis application. One feature of the dynamic container creation service 4000 is it that it provides support for "plug-in" integration such that client applications and instruments can dynamically gather information from a container type service provider, such as an analysis application, and dynamically construct a container editor 4005.

In one aspect each instrument services service provider may provide an application installer 4010 that registers a selected application for a particular instrument type. However, the service provider need not be the sole provider for this information. As an example, a selected instrument (e.g. sequencer) may populate a set of applications (e.g., a default fragment analysis application). If a new analysis application is created for the instrument platform, an application installer 4010 can be provided that utilizes the existing set of attributes independent of the instrument itself.

In one aspect, the instrument services provide an interface implementation for an application installer to register application specific container 4020 and sample definitions, an example of which is shown below:

ICFResult ICFRegisterContainerParameters(String containerDefinition) 4030;

ICFResult ICFRegisterApplicationParameters(String applicationDefinition) 4040;

In this example, the parameters, containerDefinition and applicationDefinition, may be implemented in XML format and can be validated by a suitable XML schema (see Figure 42-43). The returned ICFResult object contains ICFResult.RESULT_CODE_OK in ICFResult.getResultCode() upon successful operation.

In another aspect, instrument services may also provide an interface implementation for a type service provider to retrieve specific container and application parameters, an example of which is shown below:

ICFResult ICFGetContainerParameters() 4050;

ICFResult ICFGetApplicationParameters () 4060;

In this example, if the operation is determined to be successful, the returned ICFResult object may be designated to contain an XML string definition that was installed by the application installer using the ICFRegisterApplication call. A successful operation results in the returned ICFResult object containing an XML sample definition. The type

and content of the information provided depends on the provider's implementation and it will be appreciated that a type service provider may be part of an instrument service provider or a standalone type service provider.

In another aspect, the GUI client 4005 may use the XML container definition to construct a container creator. Figure 38 illustrates a sample snapshot 4100 from a data collection instrument. In this example, the GUI container creator may comprise attributes, such as "Container Name", "Plate ID", "Description", among others. These attributes may be extracted from the XML container definition registered by the application installer. In one aspect, a customized container editor for "Genemapper" may be constructed following execution of the "OK" button 4105. In this example the customized container editor may extract column names from an XML sample definition registered by application installer.

In another aspect, container import may be implemented as functions commonly used in instruments supported by instrument services. The following Figures 39 - 41 illustrate the "pulling" and "pushing" of a logical container to import the container into a instrument service provider 4210. As shown in Figure 39, a custom client 4205 may use the function subscribeInstrumentEvents() to prepare itself to receive information from the ICFStatusEvents() and ICFContainerStatusEvents() functions. Once ready, a custom client 4205 may then use ICFRegisterContainerProvider() call 4215 to notify the instrument service provider (ISP) 4220 that the custom client 4205 is available to provide the desired container information. This function call may further return a Boolean result to indicate an ISP acknowledgment and further indicate if the container import functionality is supported. The client 4205 may further use this result code to determine if the ISP 4220 supports the selected feature. Subsequently, the ISP 4220 may notify the Instrument to process the next available container upon receiving a "start run" command from the user interface component (not shown in Figure). In various embodiments, the ISP 4220 may receive a container barcode identifier as the return of processNextContainer() 4225. The ISP may then attempt to locate the container's information from its own database. The flowchart shown in Figure 40 illustrates details for this conditional situation.

In one aspect, if the desired information 4325 is available from the database 4330, the ISP 4220 may schedule a run 4335 for this container and perform run(s) using this container. If ISP 4220 does not find the container's information in its database and at least one custom client has registered with the ISP 4220 to provide container information, the ISP 4220 may send out a ICFContainerStatusEvents() 4340 call to the messaging service. A custom client 4205 receives this event and uses the importContainer()function to provide container information to the ISP 4220. When the ISP 4220 finishes the

scheduled run(s) 4335 for this container, a "run completed" notification may be sent in the ICFStatusEvents()function. Subsequently, the custom client 4205 may update the status for the container in its database.

Figure 41 illustrates how a custom client 4205can "push" container information to a service provider 4220 to persist in the service provider's database allowing it to be run at a later time. In the example, the custom client 4220, which may be a GUI that allows user to type or scan a container barcode or other identifier sample information, requests container information from the LIMS system 4430. Once the container information is acquired, the client 4205 may use the importContainer()function 4440 to "push" it into ISP 4220.

In one aspect, the service provider 4220 provides logical container information by implementing the aforementioned interface definition to provide synchronous service. This service is designed to provide some of the functionalities illustrated in the use case diagram (Figure 41) for integrating with a LIMS client 4430. In various embodiments, an instrument service provider 4220 may implement the interface according to the function:

ICFResult importContainer(String plateText);

Here the client may provide an XML data string (conforming to the container XML schema) as a parameter for the call. An ICFResult object containing ICFResult.RESULT_CODE_OK in ICFResult.getResultCode() may then be returned upon a successful operation The service provider may further publish an ICFContainerStatusEvents function when container information is desirable to run or execute a physical container.

Figures 42 - 46 illustrate various embodiments of XML schema for application and container definitions that may be used in implementation of the aforementioned service sets and container creation/utilization services. It will be appreciated by one of skill in the art that these schemas may be readily modified to accommodate new or different data types without departing from the scope of the present teachings.

The aforementioned description of instrument and application services desirably provides a highly configurable and readily modifiable means for control and management across different instrument and application platforms. Furthermore, these services provide for improved integration of sample identification functionality requiring little or no user input.

In one aspect, the "push" / "pull" functionality described above provides a convenient method for implementing "just in time" sample identification and information

import functionality wherein a selected instrument can obtain information for a particular sample or analysis on an as needed basis rather than storing a large amount of data describing numerous samples which may or may not be utilized. The just in time functionality helps to provide appropriate sample and analysis information on an as need basis thereby conserving system resources and providing for relatively seamless integration of instruments and applications within the LIMS environment. Additional details of the just in time functionality are described elsewhere.

Dynamic Loading of User Interface

In various embodiments, the user interface may be desirably implemented in a dynamically loaded manner such that a relatively consistent view is maintained for different instrument and application types. View consistency typically improves the user's ability to interact with the LIMS system through the user interface as the user may become accustomed to a generic application and instrument presentation wherein fields, buttons, and other information is situated in the user interface in a predicable and uniform manner.

Additionally, in conventional systems changes in an application or instrument may affect the user interface and reconfiguration of the user interface can take up much time and effort. Modifying or adding instruments or applications may significantly affect a client side viewing or interaction application and necessitate recompiling, testing, and shipping of a new/updated vendor side application which must be reinstalled on the client side.

In various embodiments, to avoid such costly and time consuming procedures, the present teachings provide means for dynamically loading of classes and components for instruments and applications. In one aspect, the client side user interface may utilize a dynamic loading functionality implemented in a programming language such as JAVA wherein various instrument and application classes are contained in jars. Additionally, it will be appreciated that the dynamic loading prototype may be used in other areas of the LIMS system outside of the user interface to provide a similar dynamic loading functionality.

In one exemplary implementation of a dynamically loaded user interface using the JAVA language, an application may be contained in a frame (JFrame) with a pane (Jpanel) contained in the frames. A panel is a view of the application and generally comprises the expected components for a selected view. In implementing the user interface, a plurality of panels may be selected from a menu list. Typically, when a user launches the user interface application, a frame may be displayed. The user them may select an application menu item and an associated panel corresponding to the desired

view will be displayed. The panel will generally contain all the necessary components for that view and correspond for example to a selected instrument or data analysis application.

Figure 47 illustrates one method 4500 for implementing a web-browser based dynamically loaded user interface implemented using the JAVA language. In state 4505, when a panel displayed in the user interface is selected, an associated panel name may be retrieved in state 4510. This name is used as an identifier to identify the appropriate jar file associated with it. For example, if a particular panel named "Run3100" is selected, a corresponding file "Run3100.jar" will be looked up in state 4515. Typically, application jars may be stored locally in a directory on the client side computer. Furthermore, a time stamp may be associated with the file which is retrieved from the file upon lookup in state 4520. This time stamp may be compared in state 4525 to that of a remotely maintained file stored on a server computer (web server). If the file does not exist or if the times stamp for the file on the client computer is outdated with respect to the remotely maintained file, the client may retrieve a new/updated file from the server computer in state 4530. Once the file has been updated, or if the file on the client computer is not outdated, the file may be executed in state 4535 and the user interface will display the appropriate information to the user via the panel.

In various embodiments, the client / server communications and interactions may occur in a web/browser based environment. Thus the client computer may implement the user interface through a web-browser and a connection to a web server may be established using an HTTP protocol using the clients's IP address as run time input. The time stamp of the panel's jar file may then be retrieved and compared with the local file's time. If the server's file is newer, it may cache/replace the old file with the new. In the final stage of user interface loading a class loader may be called to load the classes needed in the panel to thereby populate the panel with the appropriate information to be displayed to the user.

It will be appreciated that the aforementioned implementation of a dynamically generated user interface represent but one of many possible implementations the may be achieved through application of the present teachings. As such other implementations and constructions of a dynamically generated user interface are considered but other embodiments of the present teachings. Furthermore, it will be appreciated that other programming languages may be used as a means to implement the user interface and the user interface may further be constructed as a standalone application without the requirement of a web browser.

It will be appreciated that the present teachings describe an instrumentation interface application that obtains information from an instrument directory and generates a graphic user interface populated based on the information. The information for a given instrument may be formatted in a manner so as to allow the interface application to interpret the information to be displayed in a certain manner. By delegating the such details of the instrument information to a directory instead of the interface application, the interface application can be utilized more effectively for relatively large biological laboratories that have many instruments of different types. The graphic user interface generated in such a manner can be used to monitor and control a variety of different aspects of the laboratory, including instrument management, sample management, and study-related management.

Although the above-disclosed embodiments of the present invention have shown, described, and pointed out the fundamental novel features of the invention as applied to the above-disclosed embodiments, it should be understood that various omissions, substitutions, and changes in the form of the detail of the devices, systems, and/or methods illustrated may be made by those skilled in the art without departing from the scope of the present invention. Consequently, the scope of the invention should not be limited to the foregoing description, but should be defined by the appended claims.

WHAT IS CLAIMED IS:

1. A communications and control system for a biological laboratory having at least one user-interfaced client for controlling and monitoring biological assays and a plurality of instruments, each with at least one associated logical component, for obtaining identification information about biological samples, the system comprising:

a plurality of logical components for the instruments that receive commands from the laboratory management system and instruct the plurality of instruments to process at least one biological sample;

wherein the plurality of logical components include at least one sample analyzer logical component for at least one sample analyzer that receives commands from the management system such that the at least one sample analyzer is instructed by the at least one sample analyzer logical component to obtain identification information about the biological sample;

a central management component in communication with the laboratory management system, wherein the central management component identifies at least one pre-defined set of instrument instructions for controlling the operation of the plurality of instruments or for communicating with the plurality of instruments;

wherein the central management component has at least one data structure that identifies the logical location of the plurality of logical components including the at least one sample analyzer logical component; and

wherein additional instruments having associated logical components can be added to the management system by updating the data structure as to the logical location of the logical component of the additional instrument such that management component is made aware of the logical location of the logical components associated with the instrument and the additional instrument can be controlled by a user via the user-interfaced client connected to the central management component without requiring modification of the at least one pre-defined set of instructions or modification of the at least one user-interfaced client.

2. The system of Claim 1, wherein the at least one sample analyzer includes one at least one sequence analyzer or bioinformatics platform that analyze DNA, RNA or protein constituents of biological samples.

3. The system of Claim 1, wherein the plurality of logical components further includes logical components for at least one thermal cycler.

4. The system of Claim 1, wherein the plurality of logical components further includes logical components for at least one robot, the robot equipped to locate and physically transfer biological samples.

5. The system of Claim 1, wherein the plurality of logical components includes logical components for at least one of biological data processor that performs one or more of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

6. The system of Claim 1, wherein the central management component comprises:

at least one instrument type server that is associated with a specific instrument type and which responds to a set of pre-defined instrument type instructions by providing data which describe instrument parameters;

an instrument directory that contains information identifying each of the plurality of instruments, including indicators of the at least one logical component associated with each of the plurality of instruments, identifies at least one pre-defined set of instructions implemented by the logical components, and contains information identifying the type and logical address of at least one instrument type server; and

a messaging service that transmits communications between the plurality of instruments.

7. The system of Claim 6, wherein the central management component responds to a request by one of the at least one user-interfaced clients for information about a requested instrument or instrument type server with a response containing a logical address for at least one logical component associated with the requested instrument or for the instrument type server so that the management system can utilize the logical address to communicate with the logical component or instrument type server via the messaging service.

8. The system of Claim 6, wherein:

the information contained in the instrument directory identifying each of the plurality of instruments includes information identifying one or more of the following: instrument type, instrument groups, unique instrument serial numbers, instrument availability, and physical locations for instruments; and the central management component provides at least some of this information to at least one user-interfaced client in response to a request by a user-interfaced client to thereby permit a user to evaluate the available instrument in the laboratory .

9. The system of Claim 6, wherein:

the information contained in the instrument directory identifying each of the plurality of instruments includes a list of topics under which messages will be published by at least one logical component associated with each instrument;

the messaging service allows user-interfaced clients to register to receive all messages published by a logical component that correspond to topics selected by the user-interfaced clients; and

the messaging service responds to publication request for a message under an associated topic from a logical component by sending the message to each of the at least one user-interfaced clients that registered to receive messages under that topic.

10. The system of Claim 6, wherein:

the instrument directory implements the Java Naming and Directory Interface;

the messaging service implements the Java Messaging service;

each pre-defined set of instrument instructions comprises a pre-defined Java interface; and

the set of pre-defined instrument type instructions comprises a pre-defined Java interface.

11. The system of Claim 10, wherein each logical component implements at least one interface whose enumerated methods provide for one or more of the following: control for a biological instrument by implementing the pre-defined instructions, receiving instrument status information, receiving instrument history information, and transmitting container and assay information to an instrument.

12. The system of Claim 11, wherein the logical component comprises a plurality of software modules, including:

a state model module, which maintains an instrument state that corresponds to a pre-defined general instrument state model, wherein the at least one interface implementing pre-defined instructions contains a plurality of commands for causing transitions within the state model; and

a translation module, which translates commands received via the at least one interface implementing commands for causing transitions within the state model and translates these commands into commands which are understood by the instrument hardware.

13. The system of Claim 6, wherein the data which describe instrument parameters that is provided by each instrument type server includes data describing container and assay parameters and data describing manual control parameters for all instruments of the instrument type associated with the instrument type server.

14. The system of Claim 13, wherein the data which describe instrument parameters that is provided by each instrument type server comprises XML data corresponding to at least one pre-defined schema.

15. The system of Claim 13, wherein:

at least one pre-defined set of instructions implemented by a logical component associated with an instrument includes a manual control command which takes pre-defined command lines and parameters as input and directly controls the instrument hardware based on the commands and parameters; and

the pre-defined command lines and parameters are provided by an instrument type server associated with the instrument to which the manual control command is associated.

16. A control and communications system for a biological laboratory having at least one sample analyzer and biological data processor, the system comprising:

a client component having a user interface that allows a user to gain access and control or monitor the operation of biological instruments in the biological laboratory;

a plurality of instrument components associated with each of the plurality of instruments, wherein the plurality of instrument components include an instrument component for a sample analyzer that receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated sample analyzer to obtain identification information about the biological sample and an instrument component for biological data processor receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated biological data processor to analyze the identification information obtained by the sample analyzer and provides signals indicative of the analysis;

at least one directory that provides information to the client component as to the logical location of the plurality of instrument components such that the client component can determine how to access the plurality of instrument components by reference to the at least one directory wherein the at least one directory includes information about the at least one pre-set list of commands that the client component can use to send signals to the plurality of instrument components to induce the sample analyzer to obtain the identification information and to induce the biological data processor to analyze the identification information;

a messaging service that transmits messages to and from the plurality of instrument components and the client component in a standardized format wherein the messaging service, the at least one directory and the client component are configured such that additional instruments can be added to the laboratory by associating an instrument component with the instrument and updating the at least one directory with information about the logical location of the instrument component and at least one pre-set list of commands for the instrument component without requiring the modification of the at least one pre-set list of commands or modification of the client component.

17. The system of Claim 16, wherein the instrument component for a sample analyzer is a component for a sequence analyzer or bioinformatics platform that analyzes DNA, RNA or protein constituents of biological samples.

18. The system of Claim 16, wherein the instrument component for a biological data processor is a component a biological data processor that performs one or more of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

19. The system of Claim 16, wherein the plurality of instrument components further includes instrument components for at least one thermal cycler.

20. The system of Claim 16, wherein the plurality of instrument components further includes instrument components for at least one robot, the robot equipped to locate and physically transfer biological samples.

21. The system of Claim 16, further comprising at least one instrument type server that is associated with a specific instrument type and which responds to a set of pre-defined instrument type commands by providing data which describe instrument parameters; and

wherein the at least one directory contains information identifying each of the plurality of instruments and contains information identifying the type and logical address of at least one instrument type server.

22. The system of Claim 21, wherein at least one directory responds to a request by the client component for information about a requested instrument or instrument type server with a response containing a logical address for at least one instrument component associated with the requested instrument or for the instrument type server so that the client can utilize the logical address to communicate with the instrument component or instrument type server via the messaging service.

23. The system of Claim 21, wherein:

the information contained in the at least one directory identifying each of the plurality of instruments includes information identifying one or more of the following: instrument type, instrument groups, unique instrument serial numbers, instrument availability, and physical locations for instruments; and the at least one directory provides at least some of this information to the client component in response to a request by the client component to thereby permit a user to evaluate the available instrument in the laboratory .

24. The system of Claim 21, wherein:

the information contained in the at least one directory identifying each of the plurality of instruments includes a list of topics under which messages will be published by at least one instrument component associated with each instrument;

the messaging service allows user-interfaced clients to register to receive all messages published by an instrument component that correspond to a topics selected by the client component; and

the messaging service responds to publication request for a message under an associated topic from an instrument component by sending the message to each of the at least one user-interfaced clients that registered to receive messages under that topic.

25. The system of Claim 21, wherein:

the instrument directory implements the Java Naming and Directory Interface;

the messaging service implements the Java Messaging service;

each pre-set list of commands comprises a pre-defined Java interface; and

the set of pre-defined instrument type commands comprises a pre-defined Java interface.

26. The system of Claim 25, wherein each instrument component implements at least one interface whose enumerated methods provide for one or more of the following: control for a biological instrument by implementing the pre-defined instructions, receiving instrument status information, receiving instrument history information, and transmitting container and assay information to an instrument.

27. The system of Claim 26, wherein each instrument component comprises a plurality of software modules, including:

a state model module, which maintains an instrument state that corresponds to a pre-defined general instrument state model, wherein the at least one interface implementing pre-set commands contains a plurality of commands for causing transitions within the state model; and

a translation module, which translates commands received via the at least one interface implementing commands for causing transitions within the state model and translates these commands into commands which are understood by the instrument hardware.

28. The system of Claim 21, wherein the data which describe instrument parameters that is provided by each instrument type server includes data describing container and assay parameters and data describing manual control parameters for all instruments of the instrument type associated with the instrument type server.

29. The system of Claim 28, wherein the data which describe instrument parameters that is provided by each instrument type server comprises XML data corresponding to at least one pre-defined schema.

30. The system of Claim 28, wherein:

at least one pre-defined set of instructions implemented by an instrument component associated with an instrument includes a manual control command which takes pre-defined command lines and parameters as input and directly controls the instrument hardware based on the commands and parameters; and

the pre-defined command lines and parameters are provided by an instrument type server associated with the instrument to which the manual control command is associated.

31. A method of controlling the operation of a biological laboratory, the method comprising:

associating an instrument component with a biological sample analyzer such that the instrument component translates received instructions to the biological sample analyzer so that the biological sample analyzer can be instructed to capture identification information about a biological sample;

associating an instrument component with a biological data processor such that the instrument component translates received instructions to the biological data processor so that the biological data processor can be instructed to analyze identification information captured by the biological sample analyzer;

maintaining a directory of information for each of the instrument components within the laboratory such that a user interface can determine by accessing the directory which of at least one pre-set list of instructions are implemented by each instrument component so that the user interface can access an instrument to implement a process; and

adding an additional instrument into the biological laboratory by associating an instrument component to the newly added instrument and updating the

directory as to the instrument component such that the new instrument can be accessed by the user interface to implement a process without requiring modification of the user interface or to any of the at least one pre-set list of instructions.

32. The method of Claim 31, further comprising transmitting signals between the user interface and the instrument components using a pre-set list of instructions which are then translated by the instrument component and provided to the instrument to thereby control the instrument.

33. The method of Claim 32, wherein associating an instrument component comprises drafting a translation process to translate the standard set of instructions into instructions that the instrument is pre-programmed to implement.

34. The method of Claim 31, wherein the biological sample analyzer can be instructed to analyze DNA, RNA or protein constituents of biological samples.

35. The method of Claim 31, wherein the biological data process can be instructed to perform one of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

36. The method of Claim 31, further comprising maintaining at least one set of instrument type information such that the user interface can determine instrument type parameters by accessing a server where the type information is maintained through a pre-set list of instrument type commands.

37. The method of Claim 31, further comprising transmitting a signal to the user interface to indicate that an additional instrument is available in the laboratory.

38. The method of 31, further comprising:

maintaining in the directory of information for each instrument a list of topics under which message will be published by at least one instrument component associated with each instrument;

registering the user interface with a messaging service such that the user interface can receive all messages published by an instrument component that correspond to topics selected by the user interface; and

transmitting messages under a listed topic name to the messaging service so that the messages will be sent to every user interface that registered to receive messages under the listed topic.

39. The method of Claim 36, further comprising responding to a request by the user interface for information about a requested instrument or instrument type with a response containing a logical address for at least one instrument component associated with the requested instrument or with a logical address for a server containing a set of

instrument type information for the requested type so that the user interface can access the requested instrument or instrument type information through a messaging service.

40. The method of Claim 36, wherein determining instrument type parameters comprises receiving XML data corresponding to at least one pre-defined schema describing instrument type parameters.

41. A control and communications system for a biological laboratory having at least one sample analyzer and biological data processor, the system comprising:

a client component having a user interface that allows a user to gain access and control or monitor the operation of biological instruments in the biological laboratory;

a plurality of instrument components associated with each of the plurality of instruments, wherein the plurality of instrument components include an instrument component for a sample analyzer that receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated sample analyzer to obtain identification information about the biological sample and an instrument component for biological data processor receives commands selected from at least one pre-set list of commands from the control system and translates the commands into a format that induces an associated biological data processor to analyze the identification information obtained by the sample analyzer and provides signals indicative of the analysis;

at least one directory that provides information to the client component as to the logical location of the plurality of instrument components such that the client component can determine how to access the plurality of instrument components by reference to the at least one directory wherein the at least one directory includes information about the at least one pre-set list of commands that the client component can use to send signals to the plurality of instrument components to induce the sample analyzer to obtain the identification information and to induce the biological data processor to analyze the identification information;

a messaging service that transmits messages to and from the plurality of instrument components and the client component in a standardized format wherein the messaging service, the at least one directory and the client component are configured such that additional instruments can be added to the laboratory by associating an instrument component with the instrument and updating the at least one directory with information about the logical location of the instrument

component so that the client component can become aware of the additional instruments by accessing the at least one directory.

42. The system of Claim 41, wherein the instrument component for a sample analyzer is a component for a sequence analyzer or bioinformatics platform that analyzes DNA, RNA or protein constituents of biological samples.

43. The system of Claim 41, wherein the instrument component for a biological data processor is a component a biological data processor that performs one or more of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

44. The system of Claim 41, wherein the plurality of instrument components further includes instrument components for at least one thermal cycler.

45. The system of Claim 41, wherein the plurality of instrument components further includes instrument components for at least one robot, the robot equipped to locate and physically transfer biological samples.

46. The system of Claim 41, further comprising at least one instrument type server that is associated with a specific instrument type and which responds to a set of pre-defined instrument type commands by providing data which describe instrument parameters; and

wherein the at least one directory contains information identifying each of the plurality of instruments and contains information identifying the type and logical address of at least one instrument type server.

47. The system of Claim 46, wherein at least one directory responds to a request by the client component for information about a requested instrument or instrument type server with a response containing a logical address for at least one instrument component associated with the requested instrument or for the instrument type server so that the client can utilize the logical address to communicate with the instrument component or instrument type server via the messaging service.

48. The system of Claim 47, wherein the client assigns a duty of the requested instrument at a high-level without having to worry about the instrument-specific details of the instrument wherein such details are provided by the at least one directory.

49. The system of Claim 46, wherein:

the information contained in the at least one directory identifying each of the plurality of instruments includes information identifying one or more of the following: instrument type, instrument groups, unique instrument serial numbers, instrument availability, and physical locations for instruments; and the at least one directory provides at least some of this information to the client component in

response to a request by the client component to thereby permit a user to evaluate the available instrument in the laboratory .

50. The system of Claim 46, wherein:

the information contained in the at least one directory identifying each of the plurality of instruments includes a list of topics under which messages will be published by at least one instrument component associated with each instrument;

the messaging service allows user-interfaced clients to register to receive all messages published by an instrument component that correspond to a topics selected by the client component; and

the messaging service responds to publication request for a message under an associated topic from an instrument component by sending the message to each of the at least one user-interfaced clients that registered to receive messages under that topic.

51. The system of Claim 46, wherein:

the instrument directory implements the Java Naming and Directory Interface;

the messaging service implements the Java Messaging service;

each pre-set list of commands comprises a pre-defined Java interface; and

the set of pre-defined instrument type commands comprises a pre-defined Java interface.

52. The system of Claim 51, wherein each instrument component implements at least one interface whose enumerated methods provide for one or more of the following: control for a biological instrument by implementing the pre-defined instructions, receiving instrument status information, receiving instrument history information, and transmitting container and assay information to an instrument.

53. The system of Claim 52, wherein each instrument component comprises a plurality of software modules, including:

a state model module, which maintains an instrument state that corresponds to a pre-defined general instrument state model, wherein the at least one interface implementing pre-set commands contains a plurality of commands for causing transitions within the state model; and

a translation module, which translates commands received via the at least one interface implementing commands for causing transitions within the state model and translates these commands into commands which are understood by the instrument hardware.

54. The system of Claim 46, wherein the data which describe instrument parameters that is provided by each instrument type server includes data describing container and assay parameters and data describing manual control parameters for all instruments of the instrument type associated with the instrument type server.

55. The system of Claim 54, wherein the data which describe instrument parameters that is provided by each instrument type server comprises XML data corresponding to at least one pre-defined schema.

56. The system of Claim 54, wherein:

at least one pre-defined set of instructions implemented by an instrument component associated with an instrument includes a manual control command which takes pre-defined command lines and parameters as input and directly controls the instrument hardware based on the commands and parameters; and

the pre-defined command lines and parameters are provided by an instrument type server associated with the instrument to which the manual control command is associated.

57. The system of Claim 41, wherein the at least one directory is configured to include information about software applications being used in the system and wherein addition of a new application to the system can be facilitated by updating the at least one directory to include information about the new application thereby allowing the new application to be integrated into the system.

58. The system of Claim 41, wherein the system is configured to provide a hardware sharing layer that allows simultaneous connection to an instrument having a limited number of access ports by multiple clients.

59. The system of Claim 58, wherein the hardware sharing layer supports Java and non-Java based clients.

60. The system of Claim 58, wherein the hardware sharing layer supports web-browser based connectivity.

61. The system of Claim 58, wherein the hardware sharing layer allows one connected client to be informed of the activity of another connected client.

62. The system of Claim 58, wherein the hardware sharing layer is configured to provide a deliverable communications implementation in the absence of a full data collection infrastructure.

63. The system of Claim 58, wherein the hardware sharing layer is configured to be substantially independent of underlying communications layers and instrument specific implementations so as to provide for improved compatibility and flexibility.

64. A communications and control system for a biological laboratory having at least one user-interfaced client for controlling and monitoring biological assays and a plurality of instruments, each with at least one associated logical component, for obtaining identification information about biological samples, the system comprising:

a plurality of logical components for the instruments that receive commands from the laboratory management system and instruct the plurality of instruments to process at least one biological sample;

wherein the plurality of logical components include at least one sample analyzer logical component for at least one sample analyzer that receives commands from the management system such that the at least one sample analyzer is instructed by the at least one sample analyzer logical component to obtain identification information about the biological sample;

a central management component in communication with the laboratory management system, wherein the central management component identifies at least one pre-defined set of instrument instructions for controlling the operation of the plurality of instruments or for communicating with the plurality of instruments;

wherein the central management component has at least one data structure that identifies the logical location of the plurality of logical components including the at least one sample analyzer logical component; and

wherein additional instruments having associated logical components can be added to the management system by updating the data structure as to the logical location of the logical component of the additional instrument such that the management component is made aware of the logical location of the logical components associated with the instrument and the additional instrument can be controlled by a user via the user-interfaced client connected to the central management component by allowing the user-interfaced client to become aware of the additional instrument by accessing the data structure of the central management component.

65. The system of Claim 64, wherein the at least one sample analyzer includes one at least one sequence analyzer or bioinformatics platform that analyze DNA, RNA or protein constituents of biological samples.

66. The system of Claim 64, wherein the plurality of logical components further includes logical components for at least one thermal cycler.

67. The system of Claim 64, wherein the plurality of logical components further includes logical components for at least one robot, the robot equipped to locate and physically transfer biological samples.

68. The system of Claim 64, wherein the plurality of logical components includes logical components for at least one of biological data processor that performs one or more of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

69. The system of Claim 64, wherein the central management component comprises:

at least one instrument type server that is associated with a specific instrument type and which responds to a set of pre-defined instrument type instructions by providing data which describe instrument parameters;

an instrument directory that contains information identifying each of the plurality of instruments, including indicators of the at least one logical component associated with each of the plurality of instruments, identifies at least one pre-defined set of instructions implemented by the logical components, and contains information identifying the type and logical address of at least one instrument type server; and

a messaging service that transmits communications between the plurality of instruments.

70. The system of Claim 69, wherein the central management component responds to a request by one of the at least one user-interfaced clients for information about a requested instrument or instrument type server with a response containing a logical address for at least one logical component associated with the requested instrument or for the instrument type server so that the management system can utilize the logical address to communicate with the logical component or instrument type server via the messaging service.

71. The system of Claim 70, wherein the client assigns a duty of the requested instrument at a high-level without having to worry about the instrument-specific details of the instrument wherein such details are provided by the at least one directory.

72. The system of Claim 70, wherein:

the information contained in the instrument directory identifying each of the plurality of instruments includes information identifying one or more of the following: instrument type, instrument groups, unique instrument serial numbers, instrument availability, and physical locations for instruments; and the central management component provides at least some of this information to at least one user-interfaced client in response to a request by a user-interfaced client to thereby permit a user to evaluate the available instrument in the laboratory .

73. The system of Claim 70, wherein:

the information contained in the instrument directory identifying each of the plurality of instruments includes a list of topics under which messages will be published by at least one logical component associated with each instrument;

the messaging service allows user-interfaced clients to register to receive all messages published by a logical component that correspond to topics selected by the user-interfaced clients; and

the messaging service responds to publication request for a message under an associated topic from a logical component by sending the message to each of the at least one user-interfaced clients that registered to receive messages under that topic.

74. The system of Claim 70, wherein:

the instrument directory implements the Java Naming and Directory Interface;

the messaging service implements the Java Messaging service;

each pre-defined set of instrument instructions comprises a pre-defined Java interface; and

the set of pre-defined instrument type instructions comprises a pre-defined Java interface.

75. The system of Claim 74, wherein each logical component implements at least one interface whose enumerated methods provide for one or more of the following: control for a biological instrument by implementing the pre-defined instructions, receiving instrument status information, receiving instrument history information, and transmitting container and assay information to an instrument.

76. The system of Claim 75, wherein the logical component comprises a plurality of software modules, including:

a state model module, which maintains an instrument state that corresponds to a pre-defined general instrument state model, wherein the at least one interface implementing pre-defined instructions contains a plurality of commands for causing transitions within the state model; and

a translation module, which translates commands received via the at least one interface implementing commands for causing transitions within the state model and translates these commands into commands which are understood by the instrument hardware.

77. The system of Claim 70, wherein the data which describe instrument parameters that is provided by each instrument type server includes data describing

container and assay parameters and data describing manual control parameters for all instruments of the instrument type associated with the instrument type server.

78. The system of Claim 77, wherein the data which describe instrument parameters that is provided by each instrument type server comprises XML data corresponding to at least one pre-defined schema.

79. The system of Claim 77, wherein:

at least one pre-defined set of instructions implemented by a logical component associated with an instrument includes a manual control command which takes pre-defined command lines and parameters as input and directly controls the instrument hardware based on the commands and parameters; and

the pre-defined command lines and parameters are provided by an instrument type server associated with the instrument to which the manual control command is associated.

80. The system of Claim 64, wherein the at least one directory is configured to include information about software applications being used in the system and wherein addition of a new application to the system can be facilitated by updating the at least one directory to include information about the new application thereby allowing the new application to be integrated into the system.

81. The system of Claim 64, wherein the system is configured to provide a hardware sharing layer that allows simultaneous connection to an instrument having a limited number of access ports by multiple clients.

82. The system of Claim 81, wherein the hardware sharing layer supports Java and non-Java based clients.

83. The system of Claim 81, wherein the hardware sharing layer supports web-browser based connectivity.

84. The system of Claim 81, wherein the hardware sharing layer allows one connected client to be informed of the activity of another connected client.

85. The system of Claim 81, wherein the hardware sharing layer is configured to provide a deliverable communications implementation in the absence of a full data collection infrastructure.

86. The system of Claim 81, wherein the hardware sharing layer is configured to be substantially independent of underlying communications layers and instrument specific implementations so as to provide for improved compatibility and flexibility.

87. A method of controlling the operation of a biological laboratory, the method comprising:

associating an instrument component with a biological sample analyzer such that the instrument component translates received instructions to the biological sample analyzer so that the biological sample analyzer can be instructed to capture identification information about a biological sample;

associating an instrument component with a biological data processor such that the instrument component translates received instructions to the biological data processor so that the biological data processor can be instructed to analyze identification information captured by the biological sample analyzer;

maintaining a directory of the logical location of each of the instrument components within the laboratory such that a user interface can determine how to access an instrument to implement a process using the directory; and

adding an additional instrument into the biological laboratory by associating an instrument component to the newly added instrument and updating the directory as to the location of the instrument component such that the new instrument can be discovered and accessed by the user interface to implement a process by using the directory.

88. The method of Claim 87, further comprising maintaining in the directory of logical locations information for each of the instrument components within the laboratory such that a user interface can determine by accessing the directory which of at least one pre-set list of instructions are implemented by each instrument component so that the user interface can access an instrument to implement a process.

89. The method of Claim 88, wherein an instrument to be accessed during the implementation of the process is assigned to the process at the beginning of the process.

90. The method of Claim 88, wherein an instrument to be accessed during the implementation of the process is selected from one or more of similar instruments after the commencement of the process.

91. The method of Claim 90, wherein the selection of the instrument is performed just prior to the access of the instrument, thereby improving the manner in which resources are utilized.

92. The method of Claim 88, further comprising transmitting signals between the user interface and the instrument components using a pre-set list of instructions which are then translated by the instrument component and provided to the instrument to thereby control the instrument.

93. The method of Claim 80, wherein associating an instrument component comprises drafting a translation process to translate the standard set of instructions into instructions that the instrument is pre-programmed to implement.

94. The method of Claim 87, wherein the biological sample analyzer can be instructed to analyze DNA, RNA or protein constituents of biological samples.

95. The method of Claim 87, wherein the biological data process can be instructed to perform one of the following functions: single nucleotide polymorphism analysis, structural analysis, and homology analysis.

96. The method of Claim 87, further comprising maintaining at least one set of instrument type information such that the user interface can determine instrument type parameters by accessing a server where the type information is maintained through a pre-set list of instrument type commands.

97. The method of Claim 87, further comprising transmitting a signal to the user interface to indicate that an additional instrument is available in the laboratory.

98. The method of 87, further comprising:

maintaining in the directory of information for each instrument a list of topics under which message will be published by at least one instrument component associated with each instrument;

registering the user interface with a messaging service such that the user interface can receive all messages published by an instrument component that correspond to topics selected by the user interface; and

transmitting messages under a listed topic name to the messaging service so that the messages will be sent to every user interface that registered to receive messages under the listed topic.

99. The method of Claim 96, further comprising responding to a request by the user interface for information about a requested instrument or instrument type with a response containing a logical address for at least one instrument component associated with the requested instrument or with a logical address for a server containing a set of instrument type information for the requested type so that the user interface can access the requested instrument or instrument type information through a messaging service.

100. The method of Claim 99, wherein responding to the request by the user allows the user manage the instrument at a high-level without having to worry about the instrument-specific details of the instrument wherein such details are provided by the directory.

101. The method of Claim 96, wherein determining instrument type parameters comprises receiving XML data corresponding to at least one pre-defined schema describing instrument type parameters.

102. The method of Claim 87, wherein the directory is configured to include information about software applications being used and wherein addition of a new

application can be facilitated by updating the directory to include information about the new application thereby allowing the new application to be integrated.

103. A system for controlling the operation of a biological laboratory having a plurality of biological processing devices wherein at least some of the biological processing devices have interrelated operational dependence, the system comprising:

a control system that monitors and controls the operation of the plurality of biological processing devices, wherein the control system includes a standardized state logic having a plurality of standardized commands and an interrelationship database that correlates the interrelationship between a change of state of one of the biological processing devices to another of the biological processing devices, wherein the control system, in response to receiving a state change from a first biological processing device determines a desired state of a second biological processing device from the interrelationship database and sends a standardized state change command to the second biological device;

a plurality of translation components associated with the plurality of biological processing devices wherein the plurality of translation components includes a customized instruction set that instructs the biological processing device to change from a first state to a second state in response to receiving the standardized state change command from the control system;

wherein the plurality of translation components include at least one sample analyzer translation component for at least one sample analyzer wherein the at least one sample analyzer translation component receives standardized commands from the control system such that the at least one sample analyzer is instructed by the at least one sample analyzer translation component to obtain identification information about the biological sample.

104. The system of Claim 103, wherein the standardized state diagram comprises a normal state and a fault state and wherein the standardized normal and fault states for a given devices represent functionally similar device-specific normal and fault states.

105. The system of Claim 104, wherein the normal state of the standardized state diagram comprises a task state and a standby state.

106. The system of Claim 105, wherein the normal state of the standardized state diagram further comprises a completed state.

107. The system of Claim 106, wherein the control system, upon receiving the standardized completed state from the first device, determines the second device's state in response to the first device's state.

108. The system of Claim 107, wherein the control system instructs the second device to go into the task state if it is in the standby state.

109. The system of Claim 103, wherein the plurality of biological processing devices includes a sample storage device.

110. The system of Claim 103, wherein the plurality of biological processing devices includes a sample transfer device.

111. The system of Claim 110, wherein the sample transfer device comprises a robotics device.

112. The system of Claim 103, wherein the plurality of biological processing devices includes a sample multiplexing device.

113. The system of Claim 112, wherein the multiplexing device comprises a thermalcycler device.

114. The system of Claim 103, wherein the plurality of biological processing devices includes the at least one sample analyzer.

115. The system of Claim 114, wherein the at least one sample analyzer comprises a mass spectrometer.

116. The system of Claim 114, wherein the at least one sample analyzer comprises a DNA sequencing device.

117. The system of Claim 114, wherein the at least one sample analyzer comprises an electrophoresis device.

118. The system of Claim 114, wherein the at least one sample analyzer comprises an array device.

119. The system of Claim 103, wherein the plurality of biological processing devices includes a system monitoring device that monitors the operating conditions of the biological laboratory.

120. The system of Claim 103, wherein the translation component for a given biological processing device is functionally located in a front end component of the given device.

121. The system of Claim 103, wherein the translation component for a given biological processing device is functionally located in a translation repository accessible by the control system and the given biological processing device.

122. A method of controlling the operation of a plurality of biological processing devices adapted to facilitate biological assays, method comprising:

monitoring the states of the plurality of biological processing devices in a standardized format wherein the standardized states of the plurality of biological processing devices are obtained from device-specific states of the plurality of

biological processing devices by transforming the device-specific states into the standardized format;

determining a course of action representative of an interrelated functioning of the plurality of biological processing devices based at least in part on the monitored standardized states of the plurality of biological processing devices; and

transmitting a set of standardized commands representative of the course of action to the plurality of biological processing devices, wherein the standardized commands are transformed into formats recognizable by the plurality of biological processing devices.

123. The method of Claim 122, wherein monitoring the states of the plurality of biological processing devices comprises monitoring to see if the devices are in a standardized normal state or a standardized fault state.

124. The method of Claim 123, wherein the normal state comprises a task state and a standby state.

125. The method of Claim 124, wherein the normal state further comprises a completed state.

126. The method of Claim 125, wherein determining a course of action comprises receiving the completed state of a first device and in response, determining that a second device's state is to be changed based on the first device's completed state.

127. The method of Claim 126, wherein the second device's state is to be changed to the task state if it is in the standby state.

128. The method of Claim 122, wherein transmitting the standardized commands to a given device comprises transforming the standardized commands to the device-specific commands at a front end component of the given device.

129. The method of Claim 122, wherein transmitting the standardized commands to a given device comprises transforming the standardized commands to the device-specific commands at a location that is not at given device.

130. A method of controlling the operation of a biological laboratory, method comprising:

associating a translation component with an existing instrument such that the translation component translates transmitted state signals and received instructions based at least in part on the state signals so that the instrument can be instructed to facilitate capture of identification information about a biological sample being assayed;

maintaining a directory of information for each of the translation components such that a user interface can determine by accessing the directory

which instruments are available for use to determine an assay process based at least in part on the state signals and the resulting instructions; and

updating the directory whenever an instrument is either added or removed functionally from the biological laboratory such that the user interface can adjust the manner in which the state signals and instructions are implemented in the assay process based on the availability of the instruments in the biological laboratory.

131. The method of Claim 130, wherein associating the translation component with the instrument comprises configuring a front end component of the instrument to perform the translation of the states and the instructions.

132. The method of Claim 130, wherein associating the translation component with the instrument comprises configuring an algorithm not on the instrument to perform the translation of the states and the instructions.

133. The method of Claim 130, wherein maintaining the directory comprises a listing of instruments available for use.

134. The method of Claim 133, wherein the list of instruments comprises information about the instruments such as supported interface types, physical locations of the instruments, and identifiers that allow communication via a network.

135. The method of Claim 134, wherein updating the directory comprises updating the list of instruments.

136. A system for controlling the operation of a biological laboratory having a plurality of biological processing devices, the system comprising:

a control system that monitors and controls the operation of the plurality of biological processing devices, wherein the control system includes a standardized state logic having a plurality of standardized commands and a database that correlates a change of state of a biological processing device to a corresponding command among the plurality of standardized commands;

a plurality of translation components associated with the plurality of biological processing devices wherein the plurality of translation components includes a customized instruction set that instructs the biological processing device to change from a first state to a second state in response to receiving the standardized state change command from the control system;

wherein the plurality of translation components include at least one sample analyzer translation component for at least one sample analyzer wherein the at least one sample analyzer translation component receives standardized commands from the control system such that the at least one sample analyzer is

instructed by the at least one sample analyzer translation component to obtain identification information about the biological sample.

137. The system of Claim 136, wherein the standardized state diagram comprises a normal state and a fault state and wherein the standardized normal and fault states for a given devices represent functionally similar device-specific normal and fault states.

138. The system of Claim 137, wherein the normal state of the standardized state diagram comprises a task state and a standby state.

139. The system of Claim 138, wherein the normal state of the standardized state diagram further comprises a completed state.

140. The system of Claim 136, wherein the plurality of biological processing devices includes a sample storage device.

141. The system of Claim 136, wherein the plurality of biological processing devices includes a sample transfer device.

142. The system of Claim 141, wherein the sample transfer device comprises a robotics device.

143. The system of Claim 136, wherein the plurality of biological processing devices includes a sample multiplexing device.

144. The system of Claim 143, wherein the multiplexing device comprises a thermalcycler device.

145. The system of Claim 136, wherein the plurality of biological processing devices includes the at least one sample analyzer.

146. The system of Claim 145, wherein the at least one sample analyzer comprises a mass spectrometer.

147. The system of Claim 145, wherein the at least one sample analyzer comprises a DNA sequencing device.

148. The system of Claim 145, wherein the at least one sample analyzer comprises an electrophoresis device.

149. The system of Claim 145, wherein the at least one sample analyzer comprises an array device.

150. The system of Claim 136, wherein the plurality of biological processing devices includes a system monitoring device that monitors the operating conditions of the biological laboratory.

151. The system of Claim 136, wherein the translation component for a given biological processing device is functionally located in a front end component of the given device.

152. The system of Claim 136, wherein the translation component for a given biological processing device is functionally located in a translation repository accessible by the control system and the given biological processing device.

153. A system for providing a graphic user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples, the system comprising:

an information component for instruments that includes information about the plurality of the biological processing instruments, wherein the information for a given instrument comprises a list of the instrument's parameters that can be monitored and a list of the instrument's parameters that can be controlled; and

an interface application that generates the graphic user interface based on the information for one or more of the instruments wherein the interface thus generated is populated according to at least a portion of the instrument's monitor and control parameters thereby allowing the interface application to generate different interfaces based on a single adaptable template interface.

154. The system of Claim 153, wherein the graphic user interface for a given instrument allows a user to monitor and control the instrument.

155. The system of Claim 153, wherein the interface application uses the information about the plurality of instruments to generate an instrument management graphic user interface that lists the plurality of instruments available for use.

156. The system of Claim 153, further comprising an information component for samples that includes information about the plurality of biological samples wherein the samples' information component allows the interface application to generate adaptable graphic user interfaces based on the samples' information.

157. The system of Claim 156, wherein the interface application uses the information about the plurality of biological samples to generate a sample management graphic user interface that lists the samples in the biological laboratory.

158. The system of Claim 157, wherein the sample and instrument information are combined to allow the interface application to generate a study management graphic user interface.

159. The system of Claim 158, wherein the graphic user interface is adapted to allow a user to plan a manner in which data is taken during a run.

160. The system of Claim 153, further comprising an information component for an analysis module that analyzes the data from one or more biological processing instruments wherein the analysis module's information component allows the interface

application to generate adaptable graphic user interfaces based for viewing results of the analyzed data.

161. The system of Claim 153, wherein the plurality of biological processing instruments includes a sample storage device.

162. The system of Claim 153, wherein the plurality of biological processing instruments includes a sample transferring robotics device.

163. The system of Claim 153, wherein the plurality of biological processing instruments includes a thermalcycler device.

164. The system of Claim 153, wherein the plurality of biological processing instruments includes a mass spectrometer.

165. The system of Claim 153, wherein the plurality of biological processing instruments includes a sequencer device.

166. The system of Claim 153, wherein the plurality of biological processing instruments includes an electrophoresis device.

167. The system of Claim 153, wherein the plurality of biological processing instruments includes an array device.

168. The system of Claim 153, wherein the plurality of biological processing instruments includes an analysis computing device.

169. The system of Claim 153, wherein the plurality of biological processing instruments includes a data storage device.

170. The system of Claim 153, wherein the instrument component is part of an instrument directory.

171. The system of Claim 153, wherein the graphic user interface is based on a web browser.

172. The system of Claim 153, wherein the interface application is dynamically induced to generate a graphic user interface corresponding to an added instrument.

173. The system of Claim 172, wherein the dynamically generated graphic user interface corresponding to the added instrument includes one or more protocols that can be provided by the added instrument.

174. A method of generating a graphic user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples, the method comprising:

receiving a request for a graphic user interface;

obtaining information associated with the requested graphic user interface; and

generating the requested graphic user interface by populating a template graphic user interface based on the information obtained.

175. The method of Claim 174, wherein receiving the request comprises receiving a request for a graphic user interface for one or more biological processing instruments.

176. The method of Claim 175, wherein obtaining information comprises obtaining information about the one or more biological processing instruments from an instrument directory.

177. The method of Claim 174, wherein receiving the request comprises receiving a request for a graphic user interface for one or more biological samples.

178. The method of Claim 177, wherein obtaining information comprises obtaining information about the one or more biological samples from a sample list.

179. A control system for a biological laboratory having a centralized user interface that provides monitoring and coordination functionality directed towards the operation of a plurality of laboratory components, the centralized user interface comprising:

an instrument information acquisition component that acquires information about the plurality of laboratory components, wherein the information for a selected laboratory component comprises monitoring and control parameters; and

an instrument interface generation component that generates a user interface using an adaptable interface template that is populated with at least a portion of the information acquired by the instrument information acquisition component so as to provide a means for selectively generating laboratory component interfaces using the information associated with one or more of the laboratory components.

180. The system of Claim 179, wherein the adaptable interface template used to generate the user interface receives the information for a selected laboratory component and automatically associates one or more presentation fields with information directed towards instrument monitoring.

181. The system of Claim 179, wherein the adaptable interface template used to generate the user interface receives the information for a selected laboratory component and automatically associates one or more input fields with information directed towards instrument control.

182. The system of Claim 179, wherein the instrument interface generation component receives the information from the instrument information acquisition

component and determines the current state of operation for one or more selected instruments which is thereafter integrated into the user interface.

183. The system of Claim 179, wherein the instrument interface generation component receives the information from the instrument information acquisition component and determines the availability of one or more selected instruments which is thereafter integrated into the user interface.

184. The system of Claim 179, wherein the instrument interface generation component receives the information from the instrument information acquisition component and determines a current sample or data set operated on by one or more selected instruments which is thereafter integrated into the user interface.

185. The system of Claim 179, wherein the instrument information acquisition component further provides means for determining a current sample inventory which is thereafter integrated into the user interface by the instrument interface generation component.

186. The system of Claim 179, wherein the user interface is based on a web browser.

187. The system of Claim 179, wherein the instrument interface generation component is dynamically induced to generate a user interface corresponding to an added instrument.

188. The system of Claim 187, wherein the dynamically generated user interface corresponding to the added instrument includes one or more protocols that can be provided by the added instrument.

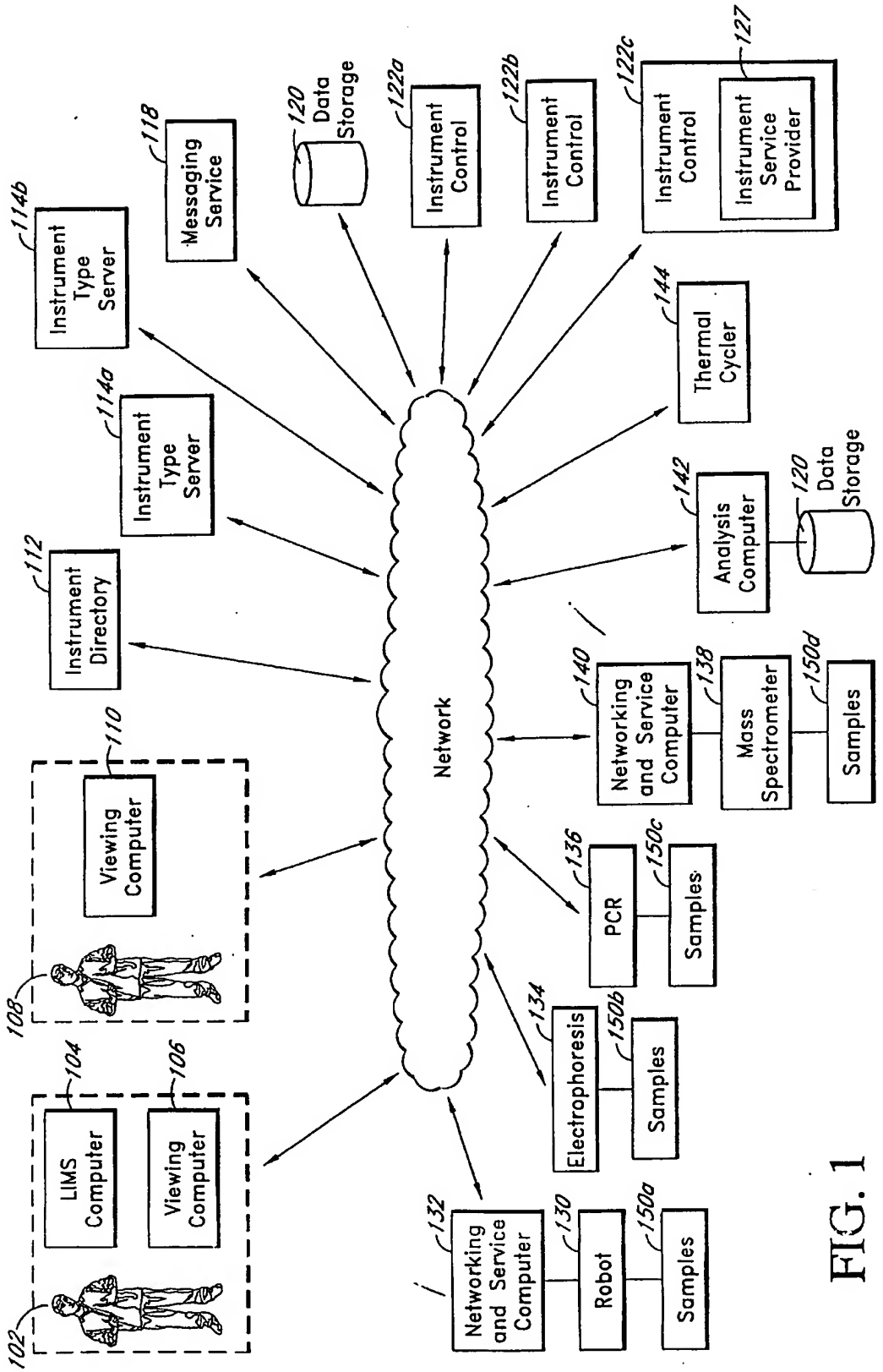

102
LIMS Computer
104
Viewing Computer
106
108
Viewing Computer
110
Instrument Directory
112
Instrument Type Server
114a
Instrument Type Server
114b
Messaging Service
118
Data Storage
120
Instrument Control
122a
Instrument Control
122b
Instrument Control
122c
Instrument Service Provider
127
Thermal Cycler
144
Analysis Computer
142
Data Storage
120
Network
Networking and Service Computer
140
Mass Spectrometer
138
Samples
150d
PCR
136
Samples
150c
Electrophoresis
134
Samples
150b
Networking and Service Computer
132
Robot
130
Samples
150a

FIG. 1

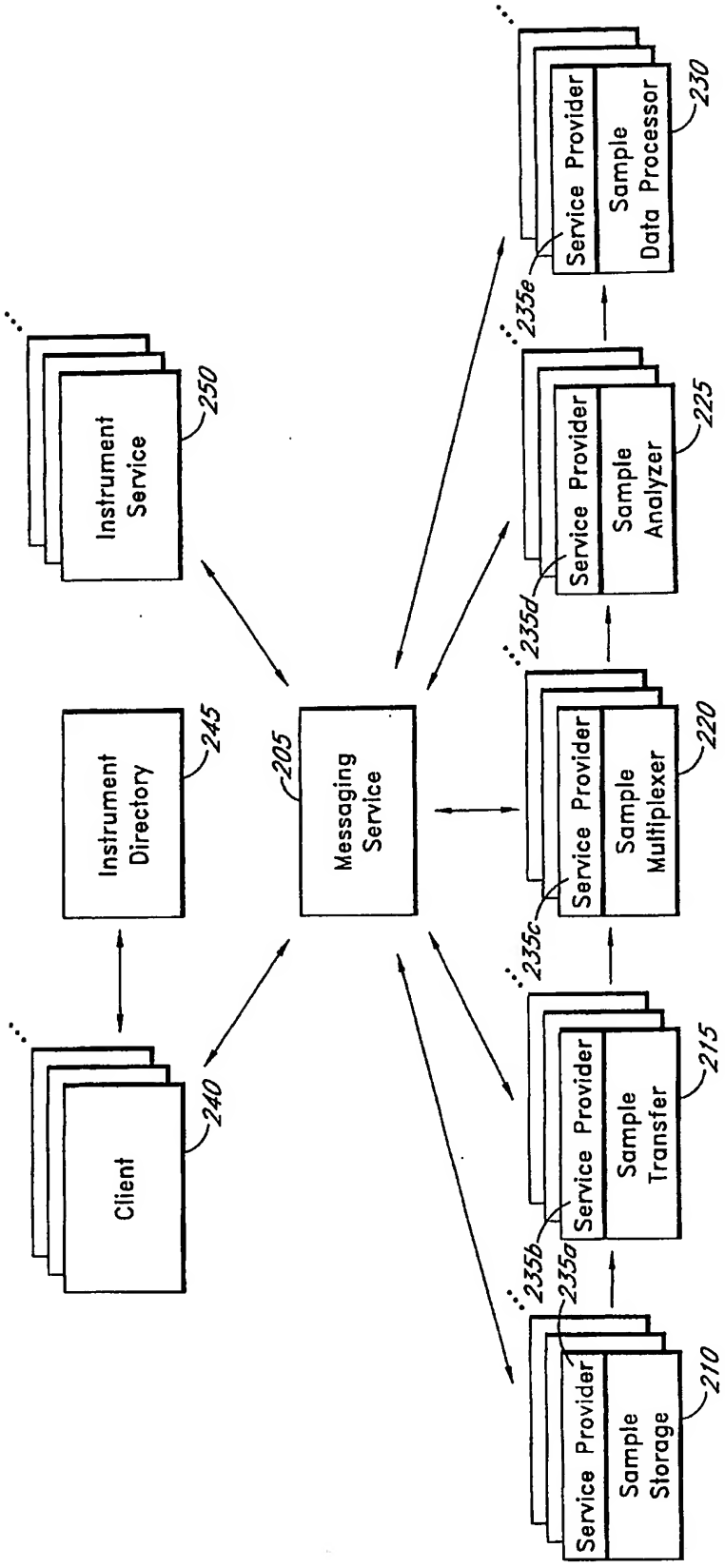

Instrument Service
250
Instrument Directory
245
Client
240
Messaging Service
205
Service Provider
Sample Storage
210
235a
Service Provider
Sample Transfer
215
235b
Service Provider
Sample Multiplexer
220
235c
Service Provider
Sample Analyzer
225
235d
Service Provider
Sample Data Processor
230
235e

FIG. 2

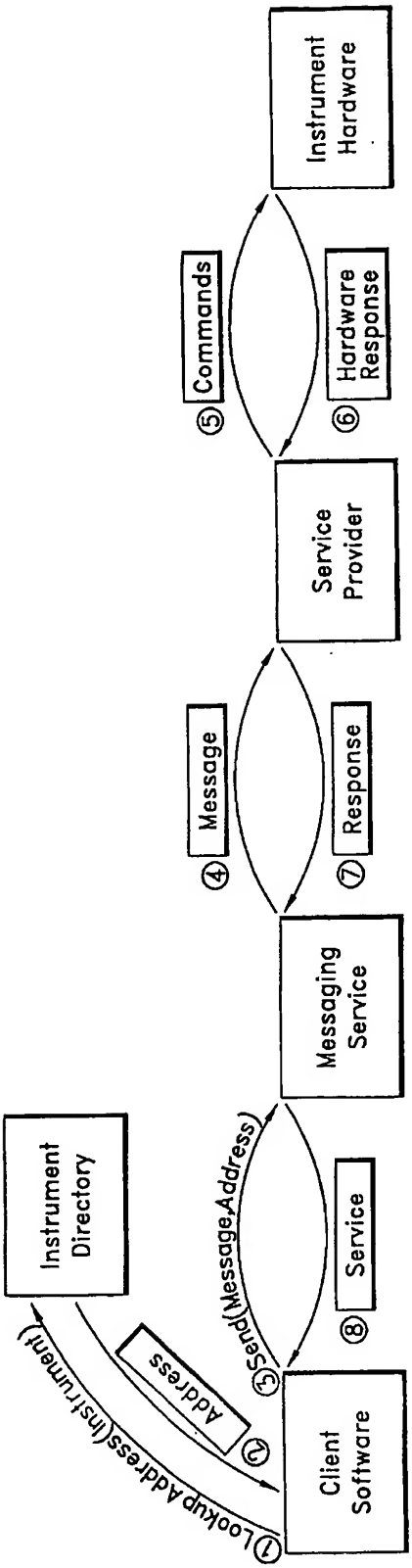

Instrument Hardware
Commands
Hardware Response
Service Provider
Message
Response
Messaging Service
Instrument Directory
Service
Send(Message,Address)
Address
LookUp Address(Instrument)
Client Software

FIG. 3A

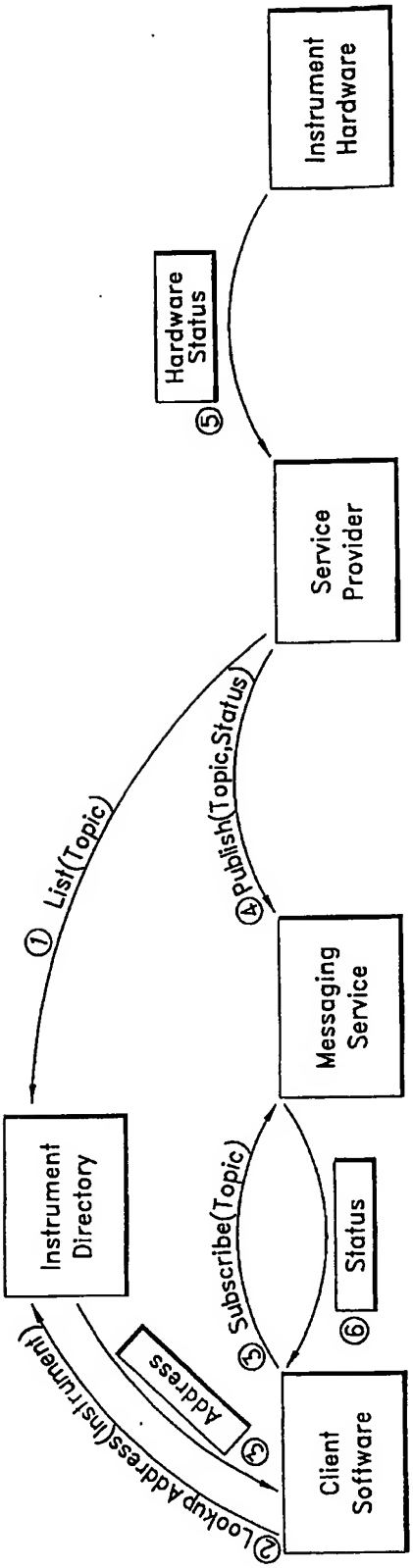
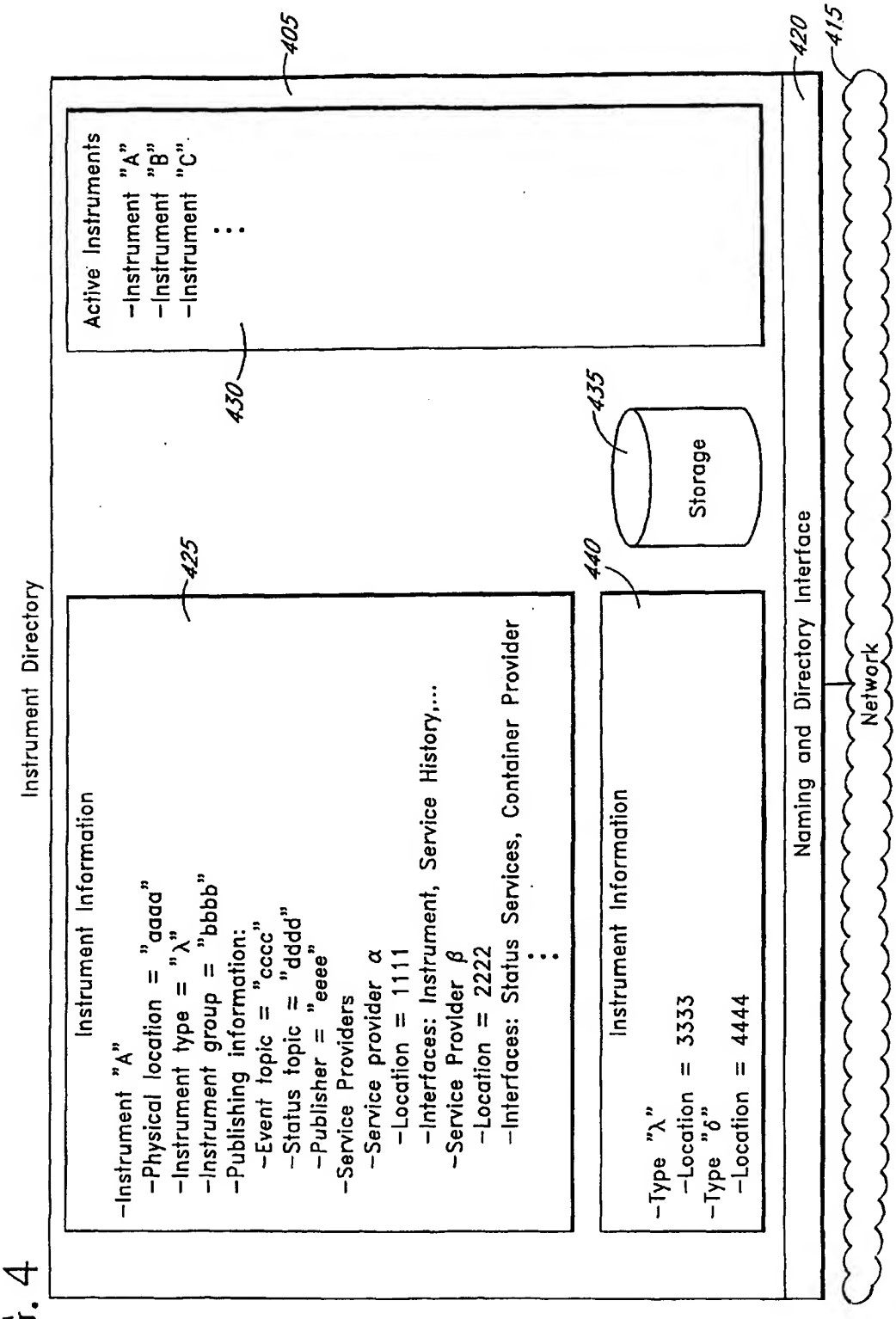

Instrument Hardware
Hardware Status
⑤
Service Provider
① List(Topic)
④ Publish(Topic,Status)
Messaging Service
Instrument Directory
③ Subscribe(Topic)
⑥ Status
② LookupAddress(Instrument)
③ Address
Client Software

FIG. 3B

FIG. 4


Instrument Directory
Instrument Information
-Instrument "A"
-Physical location = "aaaa"
-Instrument type = "λ"
-Instrument group = "bbbb"
-Publishing information:
-Event topic = "cccc"
-Status topic = "dddd"
-Publisher = "eeee"
-Service Providers
-Service provider α
-Location = 1111
-Interfaces: Instrument, Service History,...
-Service Provider β
-Location = 2222
-Interfaces: Status Services, Container Provider
425
Active Instruments
-Instrument "A"
-Instrument "B"
-Instrument "C"
405
430
Instrument Information
-Type "λ"
-Location = 3333
-Type "δ"
-Location = 4444
440
Storage
435
Naming and Directory Interface
420
Network
415

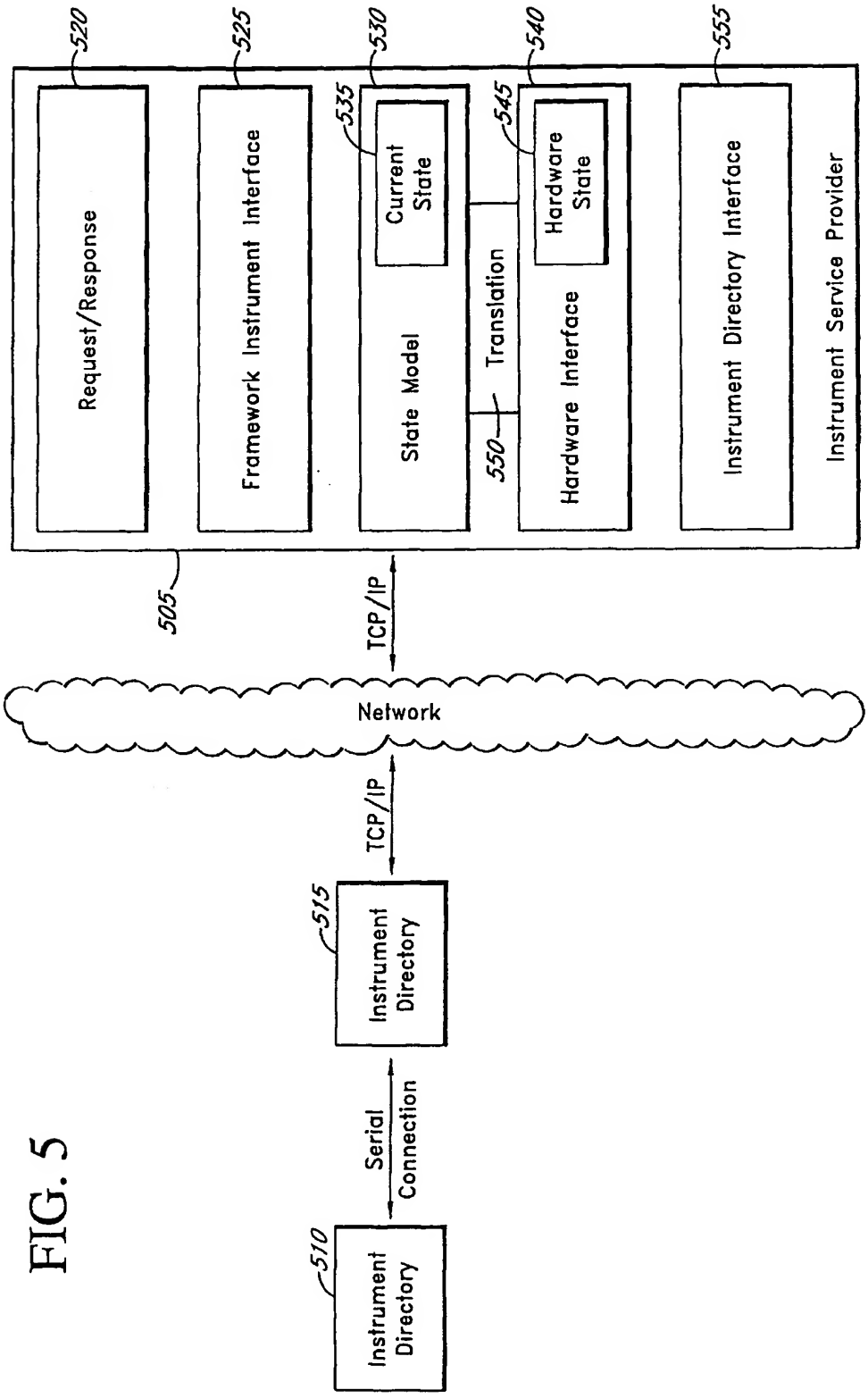
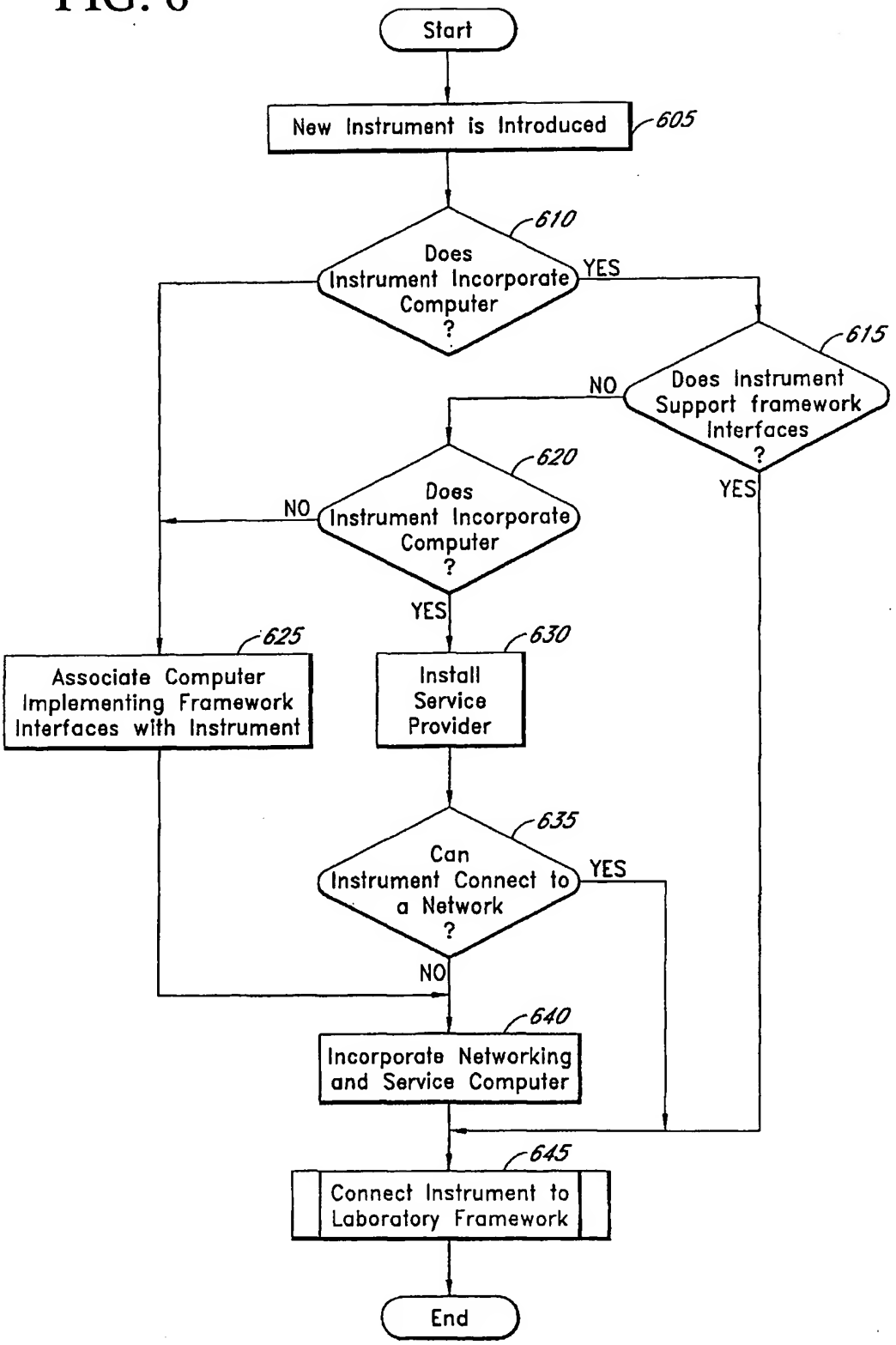

520
525
530
540
555
535
545
Request/Response
Framework Instrument Interface
Current State
State Model
Translation
Hardware State
Hardware Interface
Instrument Directory Interface
Instrument Service Provider
550
505
TCP/IP
Network
TCP/IP
515
Instrument Directory
Serial Connection
510
Instrument Directory

FIG. 5

FIG. 6


Start
New Instrument is Introduced
605
610
Does Instrument Incorporate Computer ?
YES
615
Does Instrument Support framework Interfaces ?
NO
YES
620
Does Instrument Incorporate Computer ?
NO
YES
625
Associate Computer Implementing Framework Interfaces with Instrument
630
Install Service Provider
635
Can Instrument Connect to a Network ?
YES
NO
640
Incorporate Networking and Service Computer
645
Connect Instrument to Laboratory Framework
End

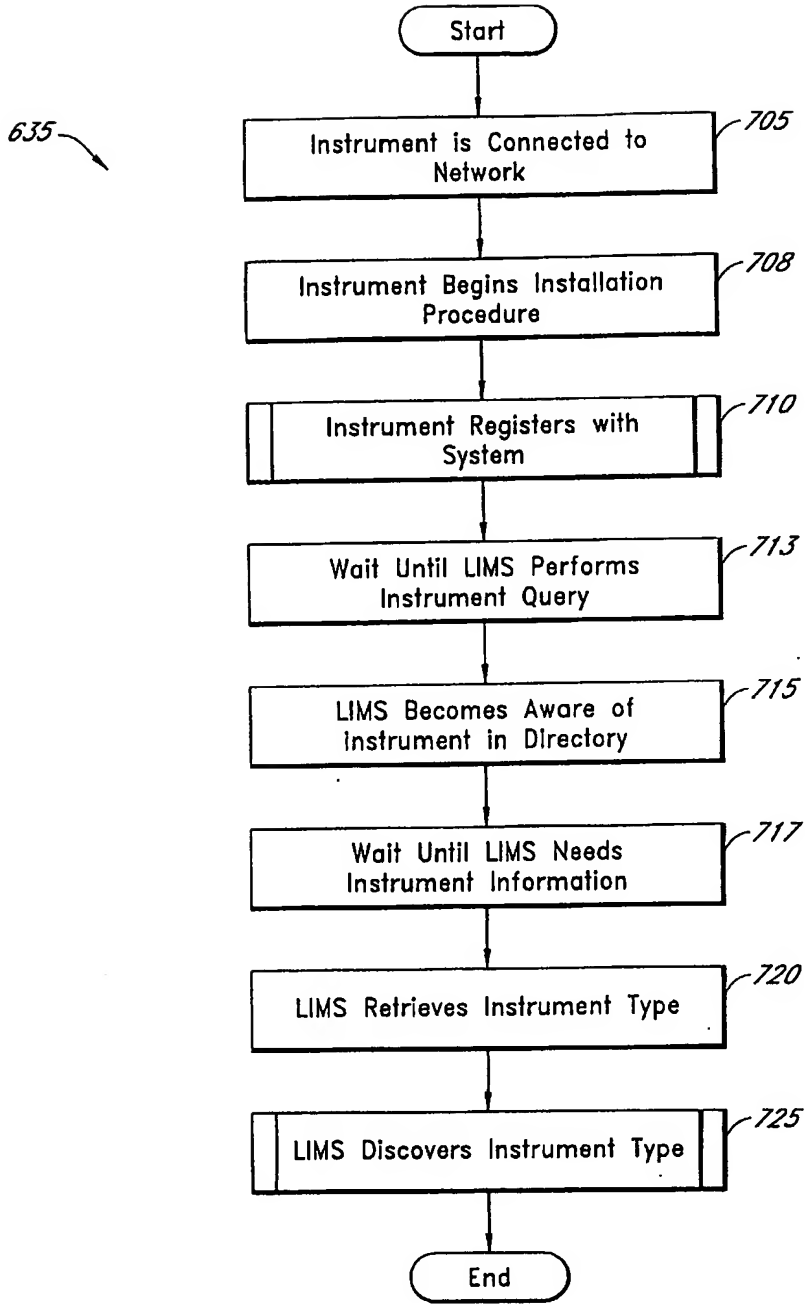
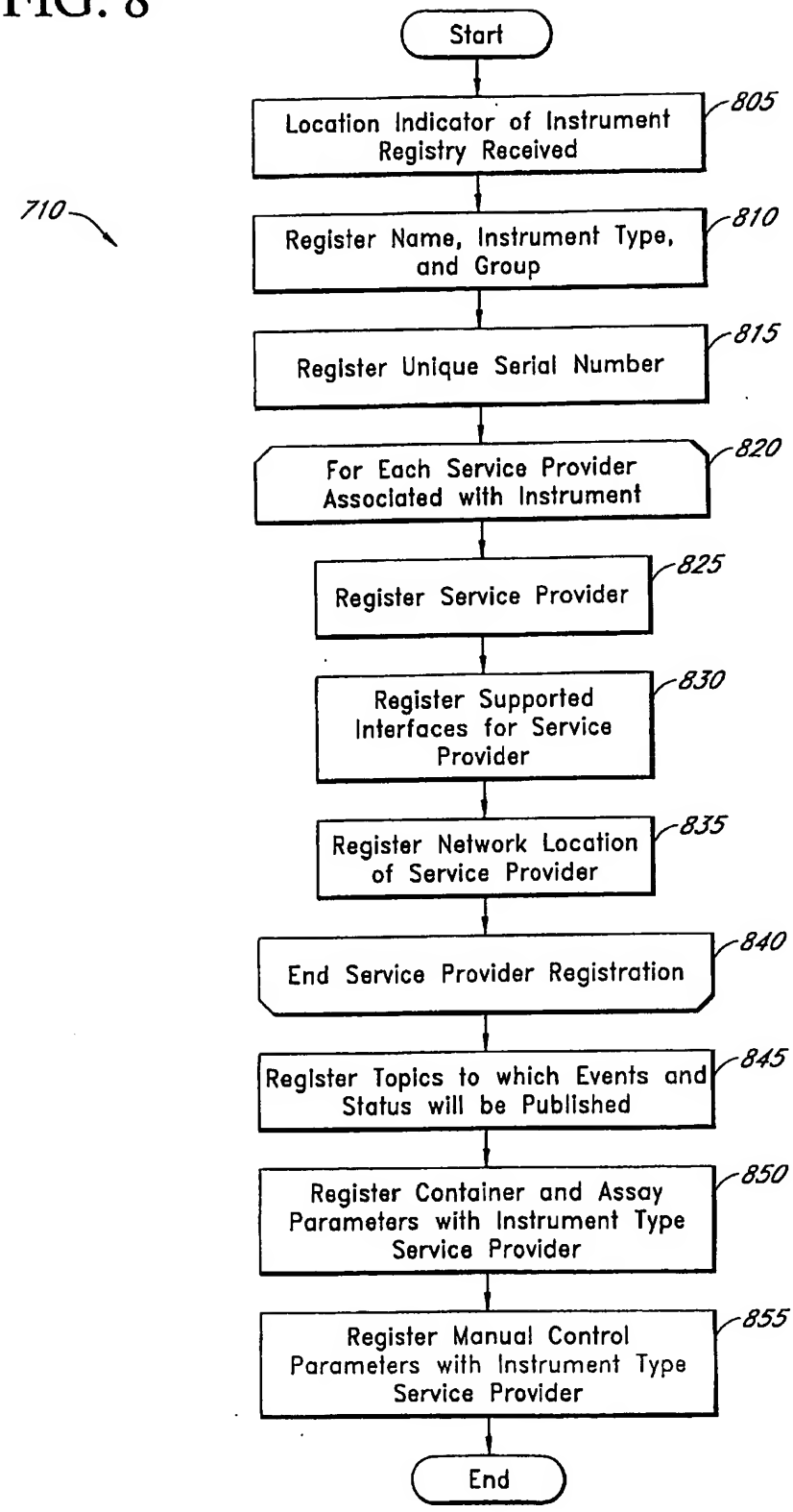

635
Start
Instrument is Connected to Network
705
Instrument Begins Installation Procedure
708
Instrument Registers with System
710
Wait Until LIMS Performs Instrument Query
713
LIMS Becomes Aware of Instrument in Directory
715
Wait Until LIMS Needs Instrument Information
717
LIMS Retrieves Instrument Type
720
LIMS Discovers Instrument Type
725
End

FIG. 7

FIG. 8


Start
Location Indicator of Instrument Registry Received
805
710
Register Name, Instrument Type, and Group
810
Register Unique Serial Number
815
For Each Service Provider Associated with Instrument
820
Register Service Provider
825
Register Supported Interfaces for Service Provider
830
Register Network Location of Service Provider
835
End Service Provider Registration
840
Register Topics to which Events and Status will be Published
845
Register Container and Assay Parameters with Instrument Type Service Provider
850
Register Manual Control Parameters with Instrument Type Service Provider
855
End

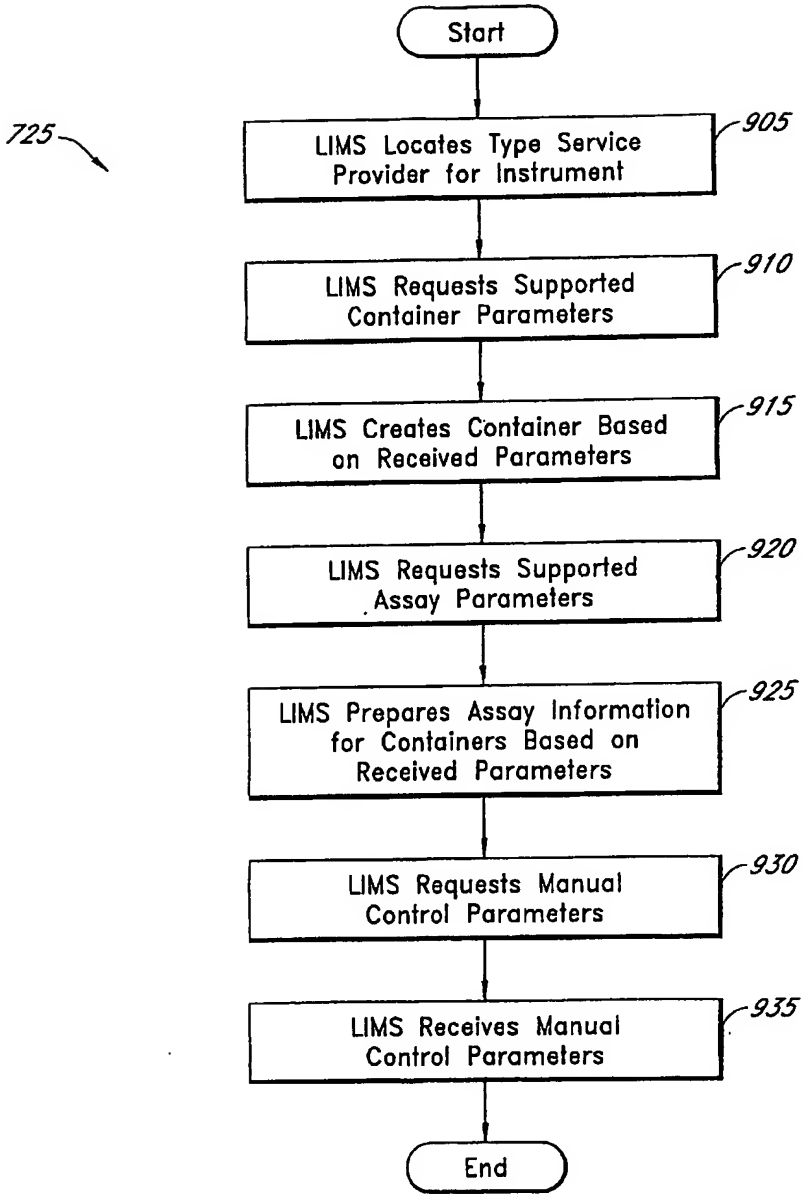

725
Start
LIMS Locates Type Service Provider for Instrument
905
LIMS Requests Supported Container Parameters
910
LIMS Creates Container Based on Received Parameters
915
LIMS Requests Supported Assay Parameters
920
LIMS Prepares Assay Information for Containers Based on Received Parameters
925
LIMS Requests Manual Control Parameters
930
LIMS Receives Manual Control Parameters
935
End

FIG. 9

FIG. 10

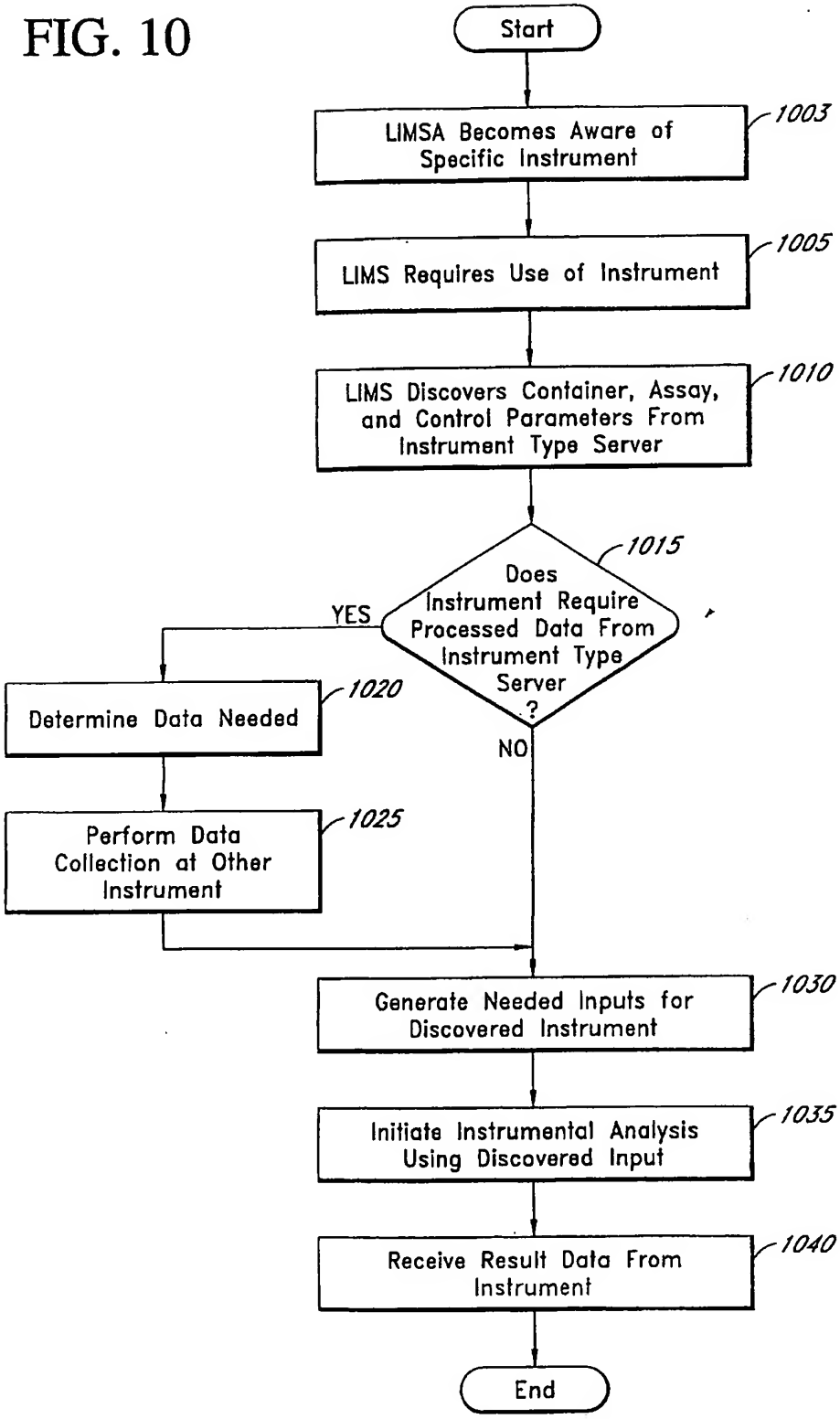
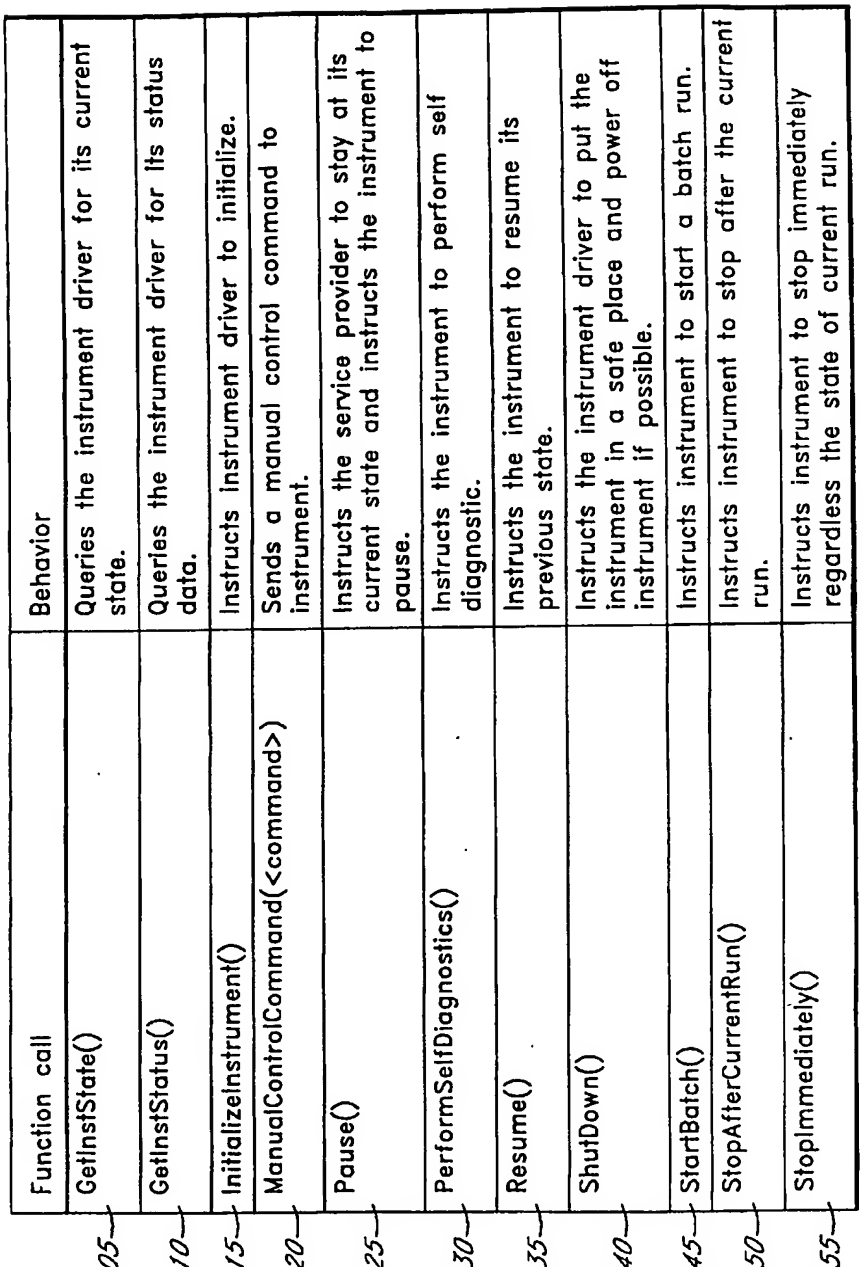

Start
LIMSA Becomes Aware of Specific Instrument
1003
LIMS Requires Use of Instrument
1005
LIMS Discovers Container, Assay, and Control Parameters From Instrument Type Server
1010
Does Instrument Require Processed Data From Instrument Type Server ?
1015
YES
NO
Determine Data Needed
1020
Perform Data Collection at Other Instrument
1025
Generate Needed Inputs for Discovered Instrument
1030
Initiate Instrumental Analysis Using Discovered Input
1035
Receive Result Data From Instrument
1040
End

FIG. 11



	Function call	Behavior
1105	GetInstState()	Queries the instrument driver for its current state.
1110	GetInstStatus()	Queries the instrument driver for its status data.
1115	InitializeInstrument()	Instructs instrument driver to initialize.
1120	ManualControlCommand(<command>)	Sends a manual control command to instrument.
1125	Pause()	Instructs the service provider to stay at its current state and instructs the instrument to pause.
1130	PerformSelfDiagnostics()	Instructs the instrument to perform self diagnostic.
1135	Resume()	Instructs the instrument to resume its previous state.
1140	ShutDown()	Instructs the instrument driver to put the instrument in a safe place and power off instrument if possible.
1145	StartBatch()	Instructs instrument to start a batch run.
1150	StopAfterCurrentRun()	Instructs instrument to stop after the current run.
1155	StopImmediately()	Instructs instrument to stop immediately regardless the state of current run.

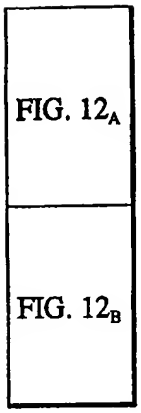


FIG. 12

FIG. 12_A

Container and Assay Parameters Example

```
- <ICFContainerParameters>
  - <ContainerNameColumn>
      <ColumnDisplayName>Container Name</ColumnDisplayName>
    </ContainerNameColumn>
  - <ContainerIDColumn>
      <ColumnDisplayName>Container Barcode ID</ColumnDisplayName>
    </ContainerIDColumn>                                          1210
  - <ContainerOwnerColumn>
      <ColumnDisplayName>Owner</ColumnDisplayName>
    </ContainerOwnerColumn>
  - <CommentColumn>
      <ColumnDisplayName>Comment</ColumnDisplayName>
    </CommentColumn>
  - <ContainerInfoEntry>
      <KeyString>PLATE_TYPE</KeyString>                           1215
      <DisplayName>Plate Type</DisplayName>
      <isInputRequired>true</isInputRequired>
    - <ValueType>
      - <List>
        - <ListItem>
            <DisplayString>96 Well Plate</DisplayString>
            <ValueString>96_WELL</ValueString>
          </ListItem>                                             1220
        - <ListItem>
            <DisplayString>384 Well Plate</DisplayString>
            <ValueString>384_WELL</ValueString>
          </ListItem>
        </List>
      </ValueType>
    </ContainerInfoEntry>
  - <ContainerInfoEntry>
      <KeyString>ASSAY_TEMPLATE</KeyString>
      <DisplayName>Assay Template</DisplayName>
      <isInputRequired>true</isInputRequired>
    - <ValueType>                                                 1225
      - <Dynamic>
        <ClassName>com.apldbio.udc.ICFImp.GetAssayType</ClassName>
        </Dynamic>
      </ValueType>
    </ContainerInfoEntry>
  + <ContainerInfoEntry>
  - <ContainerInfoEntry>
      <KeyString>PLATE_SEAL</KeyString>
      <DisplayName>Plate Sealing</DisplayName>
      <isInputRequired>true</isInputRequired>
    - <ValueType>                                                 1230
      - <List>
        - <ListItem>
```

FIG. 12_B

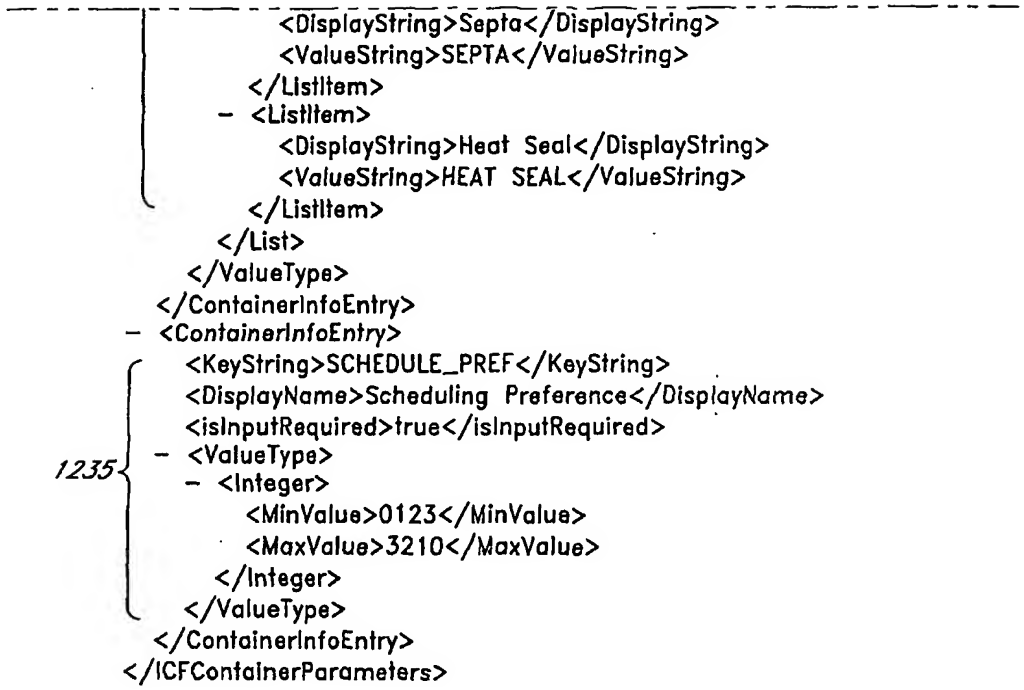

1235
<DisplayString>Septa</DisplayString>
<ValueString>SEPTA</ValueString>
</ListItem>
<ListItem>
<DisplayString>Heat Seal</DisplayString>
<ValueString>HEAT SEAL</ValueString>
</ListItem>
</List>
</ValueType>
</ContainerInfoEntry>
<ContainerInfoEntry>
<KeyString>SCHEDULE_PREF</KeyString>
<DisplayName>Scheduling Preference</DisplayName>
<isInputRequired>true</isInputRequired>
<ValueType>
<Integer>
<MinValue>0123</MinValue>
<MaxValue>3210</MaxValue>
</Integer>
</ValueType>
</ContainerInfoEntry>
</ICFContainerParameters>

FIG. 13

Container Example

```
- <ICFContainer>
    <ContainerName>GM Plate 1</ContainerName>
    <ContainerID>AB154234</ContainerID>
    <Owner>Kai Yung</Owner>                                        1305
    <Comment>This is a demo plate to demonstrate the input format for a
             IS container</Comment>
    <PlateType>96-Well</PlateType>
  - <ContainerAttribute>                                           1310
      <Key>PlateSealing</Key>
      <Value>Heat Seal</Value>
    </ContainerAttribute>
  - <ContainerAttribute>                                           1315
      <Key>SchedulingPref</Key>
      <Value>01234</Value>
    </ContainerAttribute>
  - <AssayData>
      <AssayTemplate>GM GenoTyping</AssayTemplate>
      <AssayInstance>Kai.Laptop 1</AssayInstance>
    - <SampleData>                                                 1320
        <Name>Sample 1</Name>
        <TrackingID>123456</TrackingID>
        <WellName>A1</WellName>
        <Comment>This is sample 1</Comment>
      - <SampleAttribute>
          <Key>ResultsGroup</Key>
          <Value>DefaultResultsGroup</Value>
        </SampleAttribute>
      - <SampleAttribute>
          <Key>InstrumentProtocol</Key>
          <Value>3730xlProtocol1</Value>
        </SampleAttribute>
      </SampleData>
    - <SampleData>
        <Name>Sample 2</Name>
        <TrackingID>123457</TrackingID>
        <WellName>A2</WellName>
        <Comment>This is sample 2</Comment>
      - <SampleAttribute>
          <Key>ResultsGroup</Key>
          <Value>DefaultResultsGroup</Value>
        </SampleAttribute>
      - <SampleAttribute>
          <Key>InstrumentProtocol</Key>
          <Value>3730xlProtocol3</Value>
        </SampleAttribute>
      </SampleData>
    </AssayData>
</ICFContainer>
```

FIG. 14

Manual Control Parameters Example

```
- </manual-control-commands>
  + <group>
  - <group>
1405 — <name>Electrophoresis</name>
    + <command>
    - <command>
1410 — <name>Set electrophoresis power supply</name>
1415 — <exec-string>EPS #</exec-string>
1420 — <comment>Turns the power supply on or off. Capillary ends
                should be in buffer.</comment>
      <visible>true</visible>
    - <param>
        <classification>LIST</classification>
        <default>On</default>
      - <range>
1425  - <value-list>
            <value>On</value>
            <value>Off</value>
          </value-list>
        </range>
      </param>
    </command>
  - <command>
      <name>Set electrophoresis voltage</name>
      <exec-string>EPS:VOLT:SETT #</exec-string>
      <comment>Sets voltage from 0 to 15kV. Capillary ends should be
               in buffer.</comment>
      <visible>true</visible>
    - <param>
        <classification>FLOAT</classification>
1445    <default>1.0</default>
      - <range>
          <min>0.0</min>
          <max>15.0</max>
          <unit>kV</unit>
        </range>
      </param>
    </command>
  - <command>
      <name>Read electrophoresis voltage</name>
1460  <exec-string>EPS:VOLT:READ?</exec-string>
      <comment>Reads the present power supply setting.</comment>
      <visible>true</visible>
    </command>
  </group>
</manual-control-commands>
```

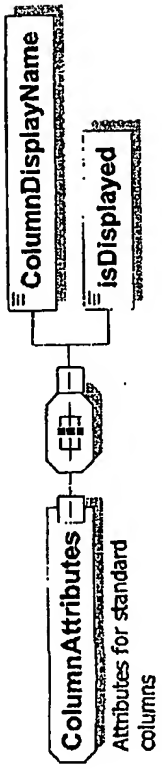

ColumnDisplayName
isDisplayed
ColumnAttributes
Attributes for standard columns

FIG. 15

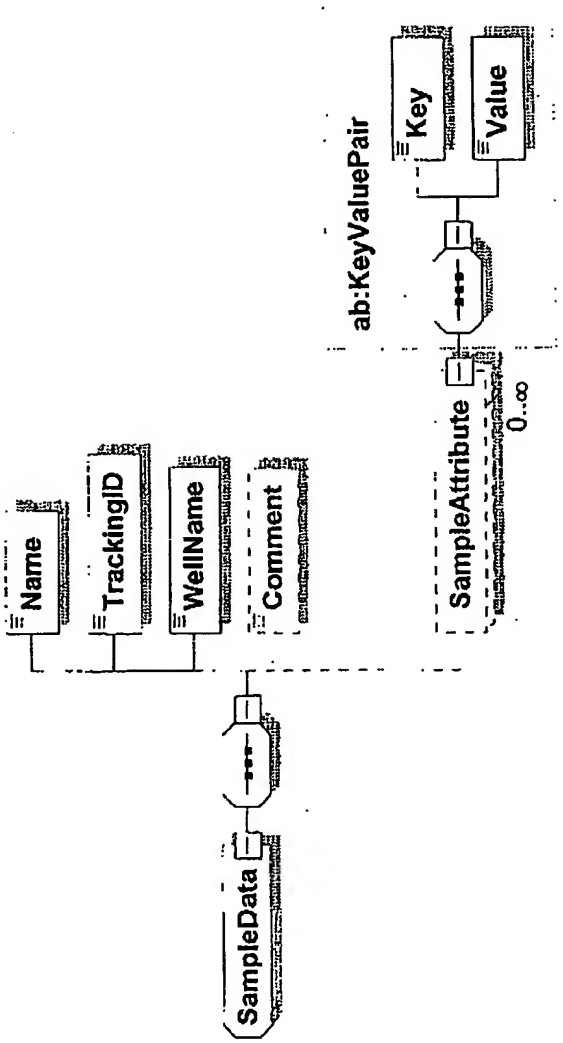

SampleData
Name
TrackingID
WellName
Comment
SampleAttribute
0..∞
ab:KeyValuePair
Key
Value

FIG. 16

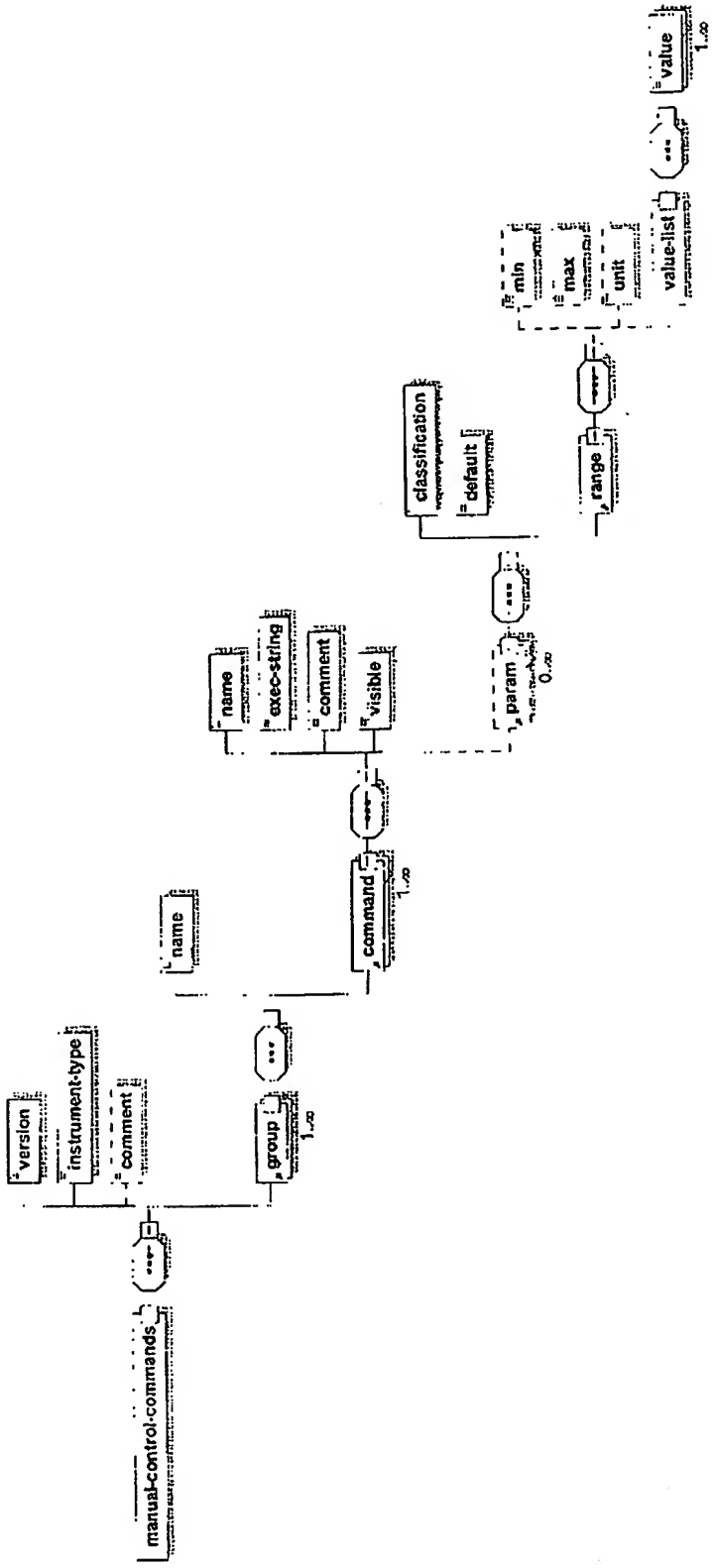

manual-control-commands
version
instrument-type
comment
group
1..∞
name
command
1..∞
name
exec-string
comment
visible
param
0..∞
classification
default
range
min
max
unit
value-list
value
1..∞

FIG. 17

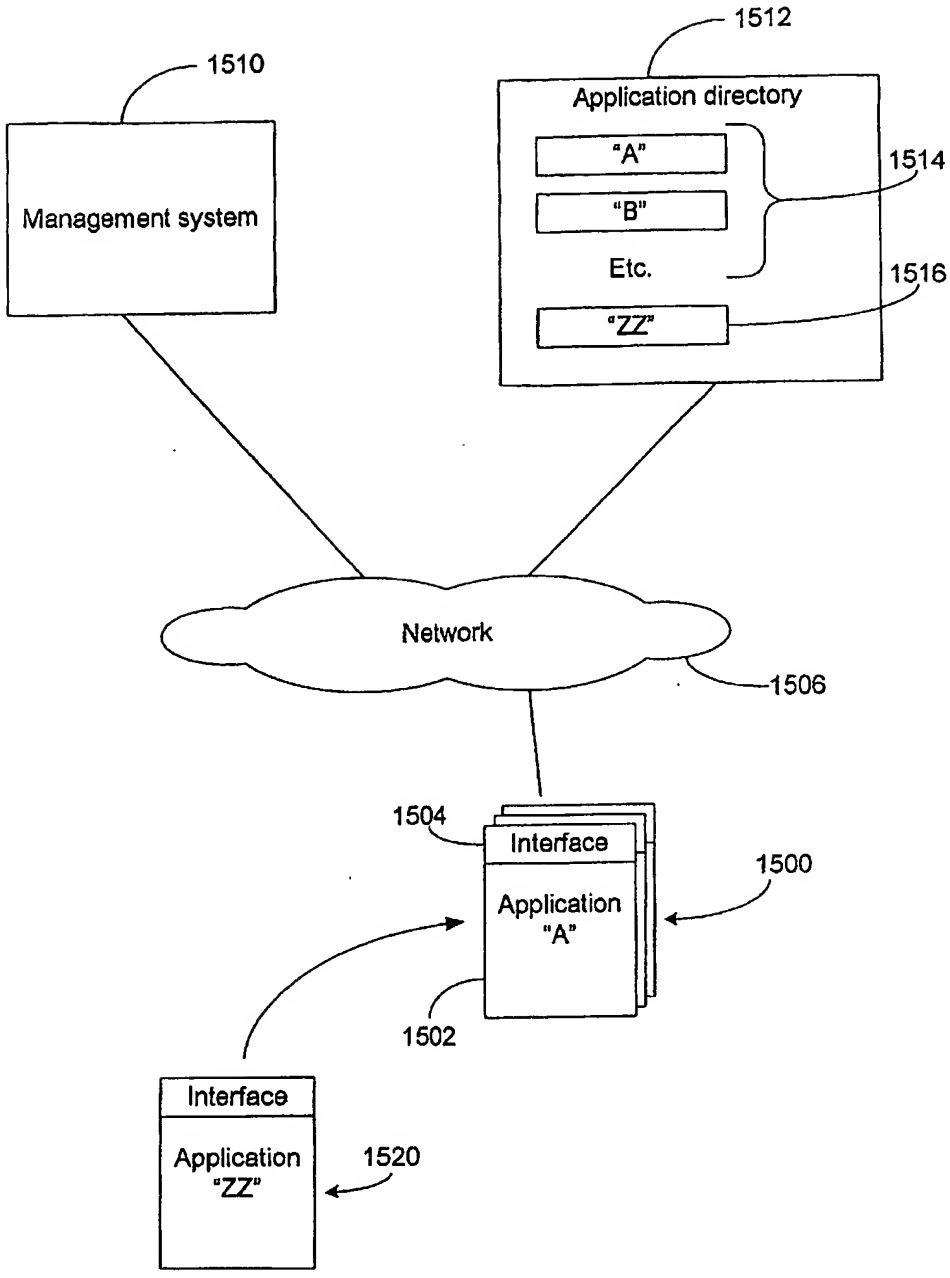

1512
1510
Application directory
"A"
"B"
Etc.
"ZZ"
1514
1516
Management system
Network
1506
1504
Interface
Application
"A"
1500
1502
Interface
Application
"ZZ"
1520

FIG. 18

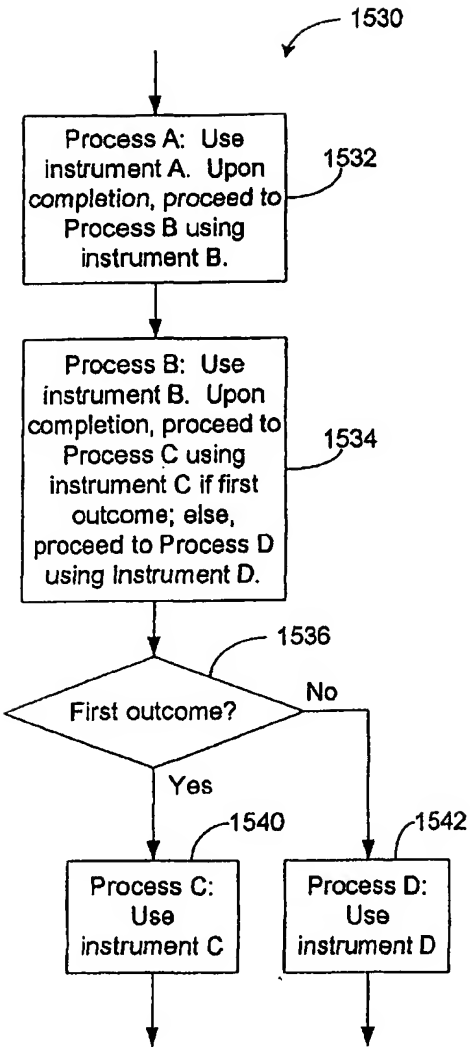


FIG. 19A

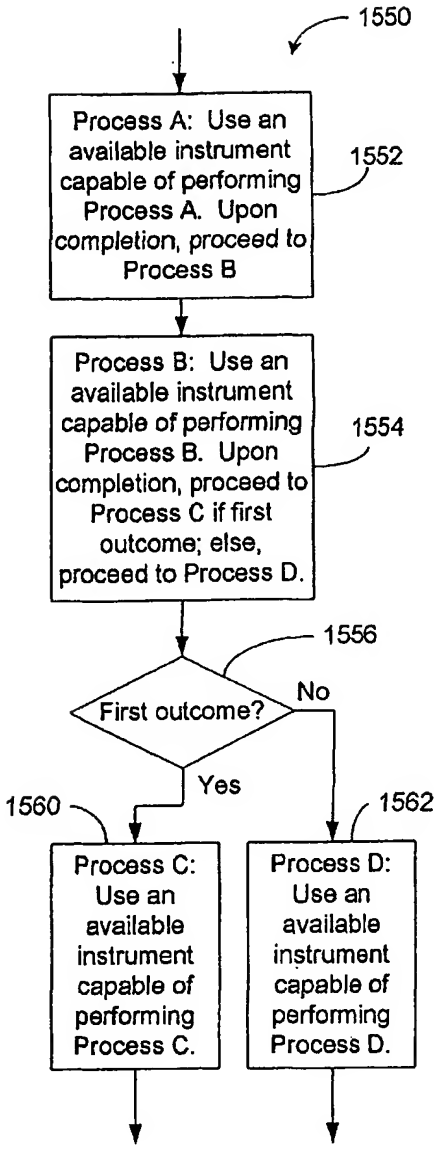


FIG. 19B

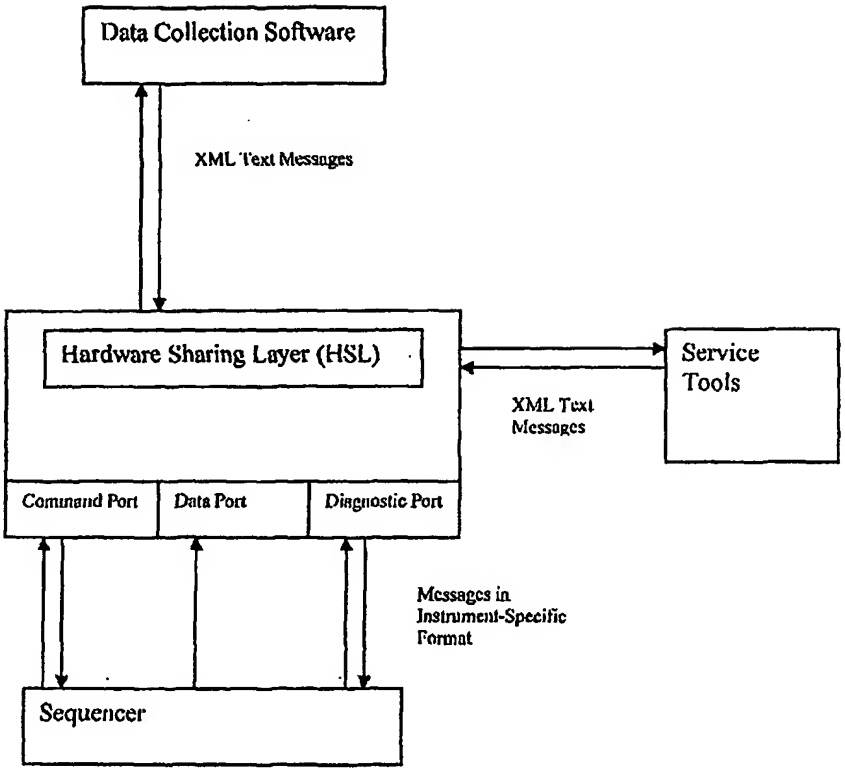

Data Collection Software
XML Text Messages
Hardware Sharing Layer (HSL)
Service Tools
XML Text Messages
Command Port
Data Port
Diagnostic Port
Messages in Instrument-Specific Format
Sequencer

Fig. 19C

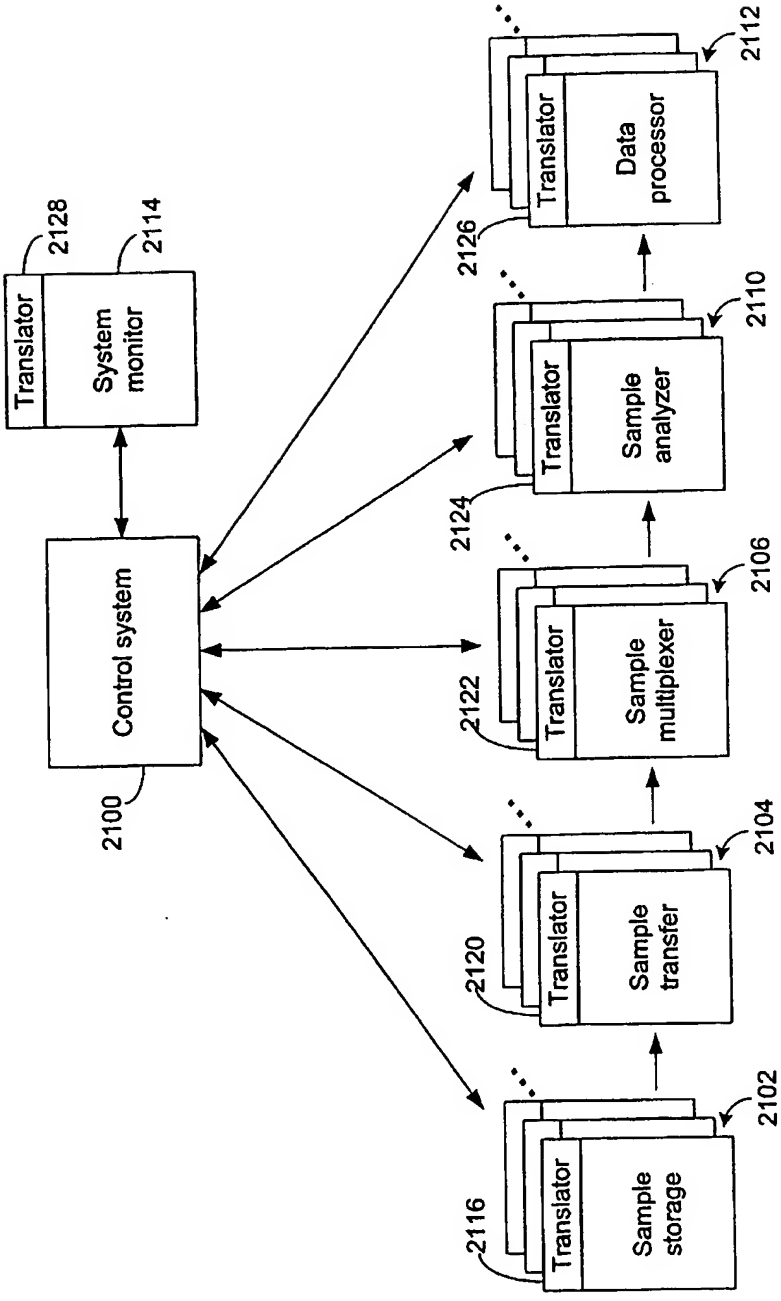

Control system
2100
Translator
System monitor
2128
2114
Translator
Sample storage
2116
2102
Translator
Sample transfer
2120
2104
Translator
Sample multiplexer
2122
2106
Translator
Sample analyzer
2124
2110
Translator
Data processor
2126
2112

FIG. 20A

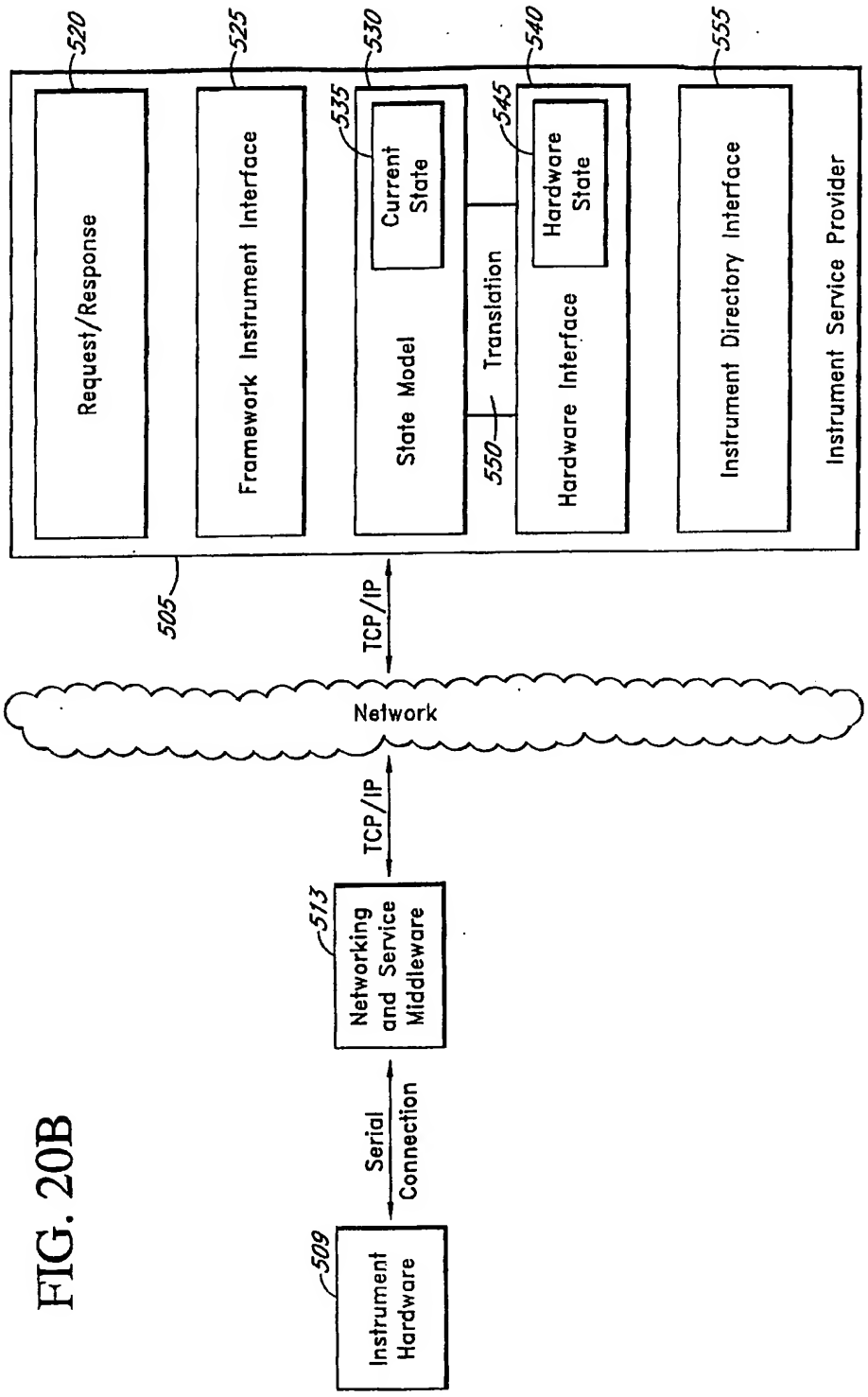

520
525
530
540
555
535
545
Request/Response
Framework Instrument Interface
Current State
State Model
Translation
Hardware State
Hardware Interface
Instrument Directory Interface
Instrument Service Provider
550
505
TCP/IP
Network
TCP/IP
513
Networking and Service Middleware
Serial Connection
509
Instrument Hardware

FIG. 20B

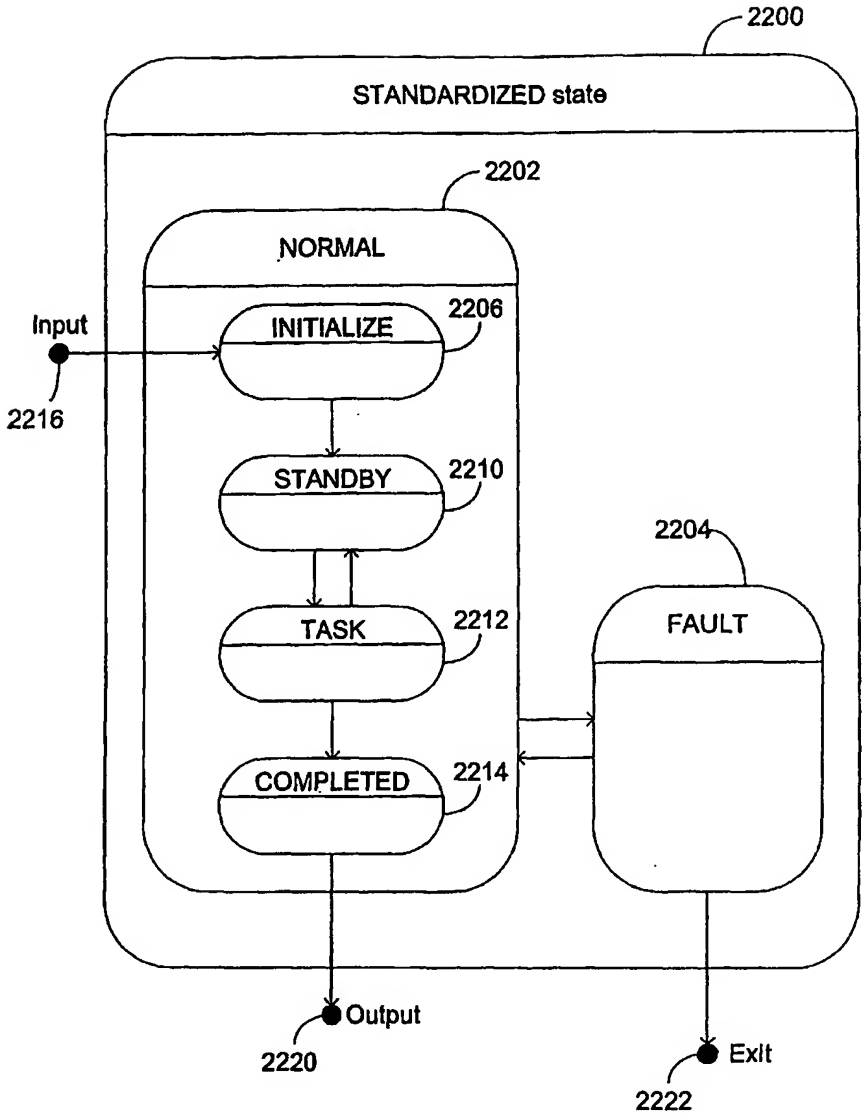

2200
STANDARDIZED state
2202
NORMAL
Input
2216
INITIALIZE
2206
STANDBY
2210
TASK
2212
COMPLETED
2214
2204
FAULT
Output
2220
Exit
2222

FIG. 20C

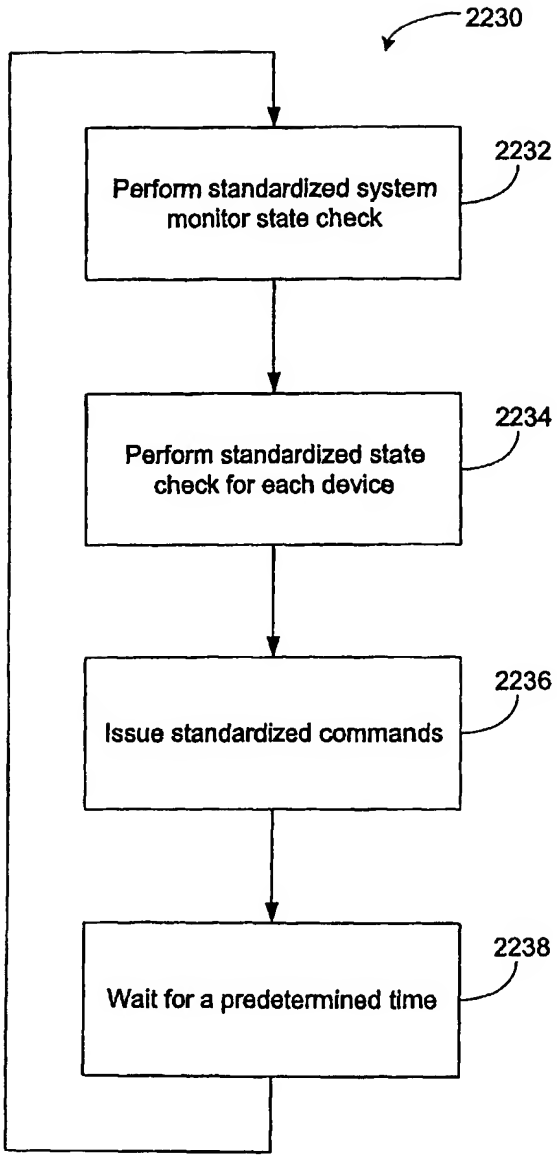

2230
Perform standardized system monitor state check
2232
Perform standardized state check for each device
2234
Issue standardized commands
2236
Wait for a predetermined time
2238

FIG. 21

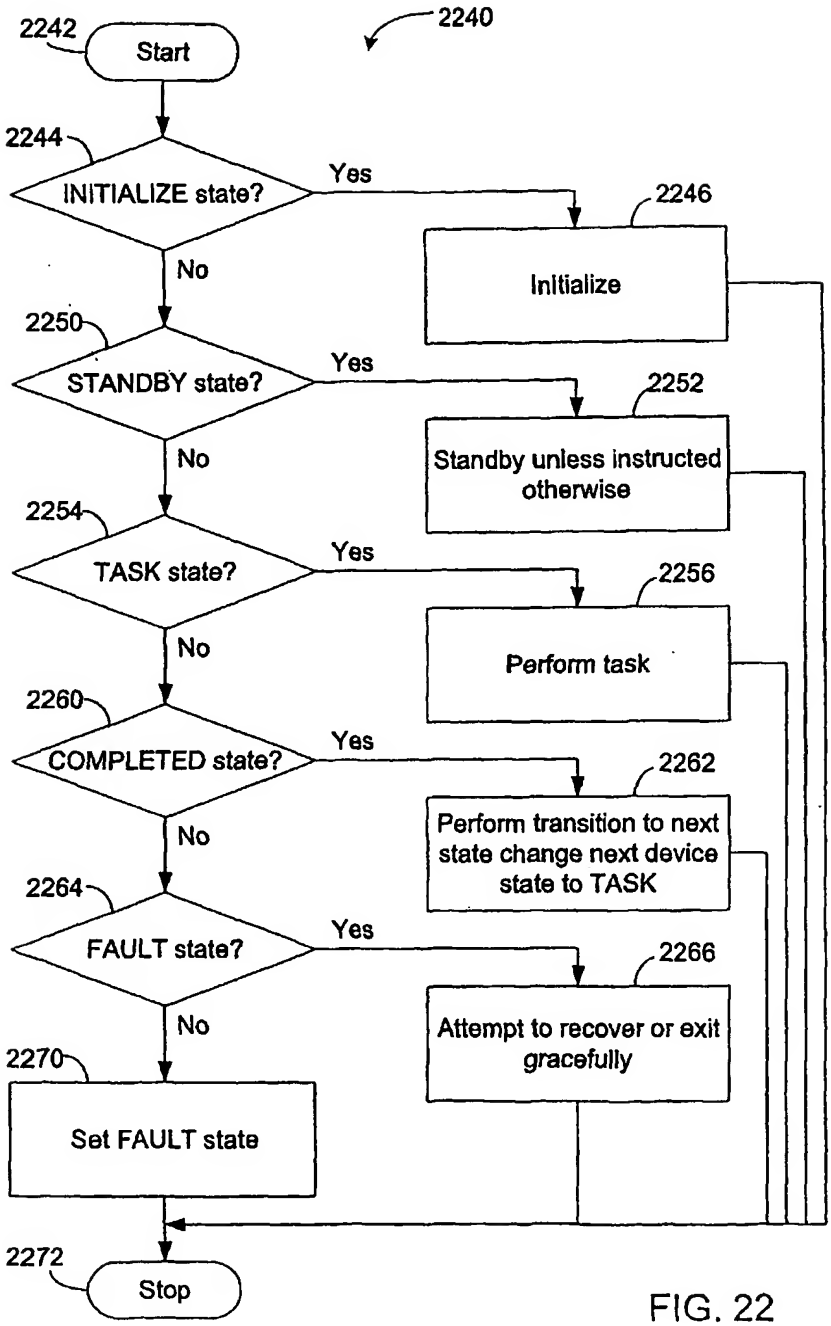

2240
2242
Start
2244
INITIALIZE state?
Yes
2246
Initialize
No
2250
STANDBY state?
Yes
2252
Standby unless instructed otherwise
No
2254
TASK state?
Yes
2256
Perform task
No
2260
COMPLETED state?
Yes
2262
Perform transition to next state change next device state to TASK
No
2264
FAULT state?
Yes
2266
Attempt to recover or exit gracefully
No
2270
Set FAULT state
2272
Stop

FIG. 22

2274

Device	If in INITIALIZE state	If in STANDBY state	If in TASK state	If in COMPLETED state	If in FAULT state
System monitor	Initialize if not initialized	Continue standby	Continue system monitoring	N/A	Halt system
Sample storage	Initialize if not initialized	Continue standby	Continue sample storage	Change Robotics to TASK	Attempt to recover or exit gracefully
Sample transfer	Initialize if not initialized	Continue standby	Continue sample transport	Change Multiplexer to TASK	Attempt to recover or exit gracefully
Sample multiplexer	Initialize if not initialized	Continue standby	Continue sample multiplexing	Change Analyzer to TASK	Attempt to recover or exit gracefully
Sample analyzer	Initialize if not initialized	Continue standby	Continue sample analysis	Change Data processor to TASK	Attempt to recover or exit gracefully
Data processor	Initialize if not initialized	Continue standby	Continue data processing	Halt system	Attempt to recover or exit gracefully

FIG. 23

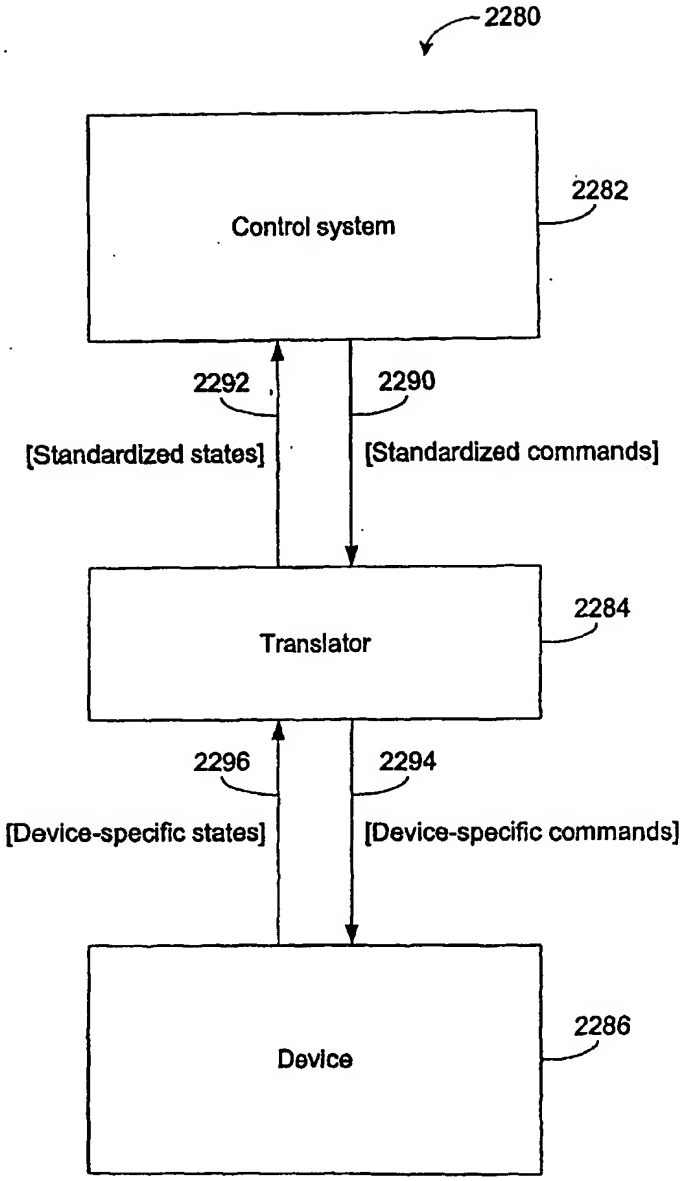

2280
Control system
2282
2292
2290
[Standardized states]
[Standardized commands]
Translator
2284
2296
2294
[Device-specific states]
[Device-specific commands]
Device
2286

FIG. 24A

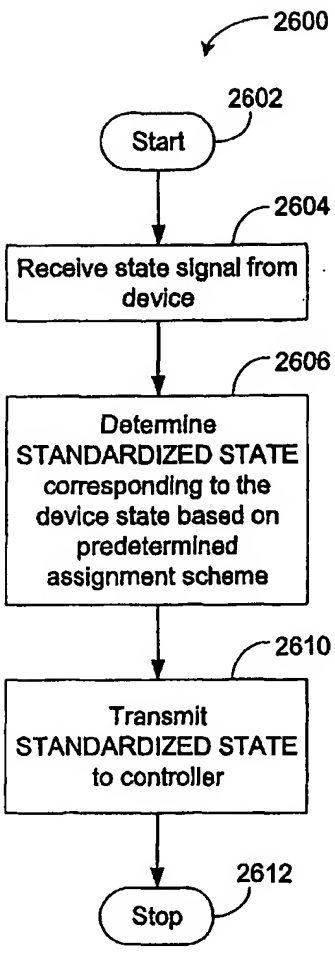


FIG. 24B

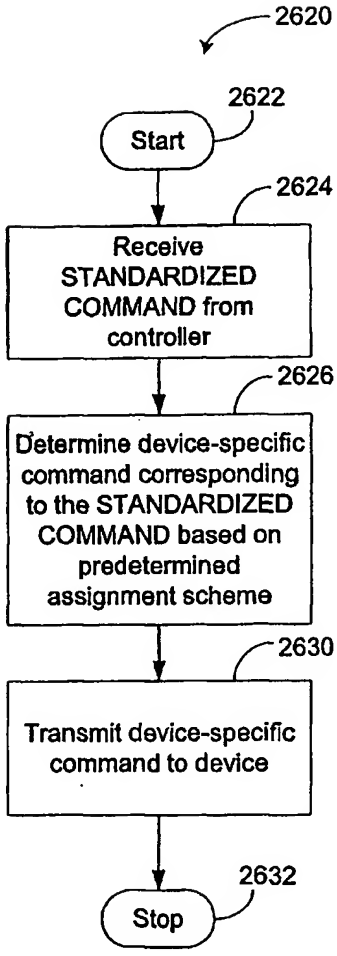


FIG. 24C

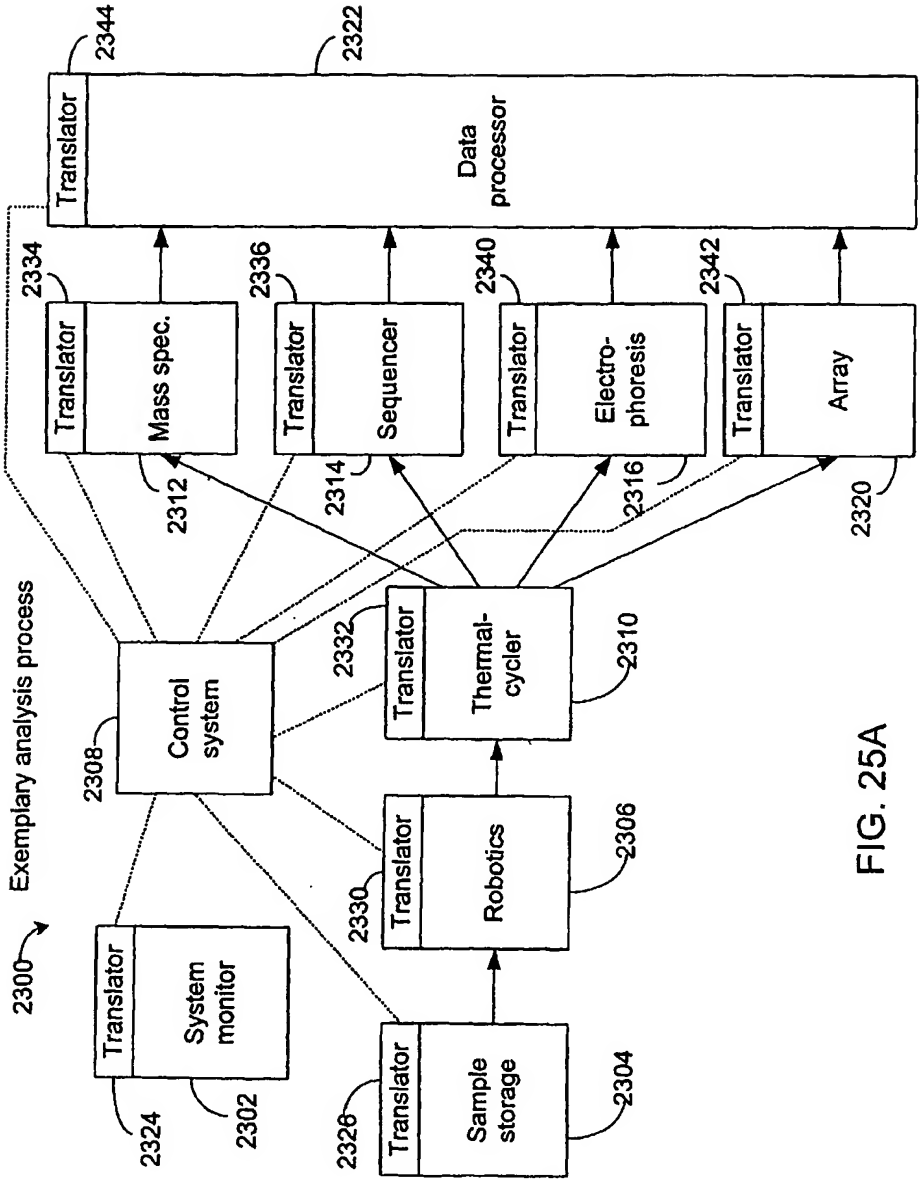

Exemplary analysis process
2300
Control system
2308
Translator
System monitor
2324
2302
Translator
Sample storage
2326
2304
Translator
Robotics
2330
2306
Translator
Thermal-cycler
2332
2310
Translator
Mass spec.
2334
2312
Translator
Sequencer
2336
2314
Translator
Electro-phoresis
2340
2316
Translator
Array
2342
2320
Translator
Data processor
2344
2322

FIG. 25A

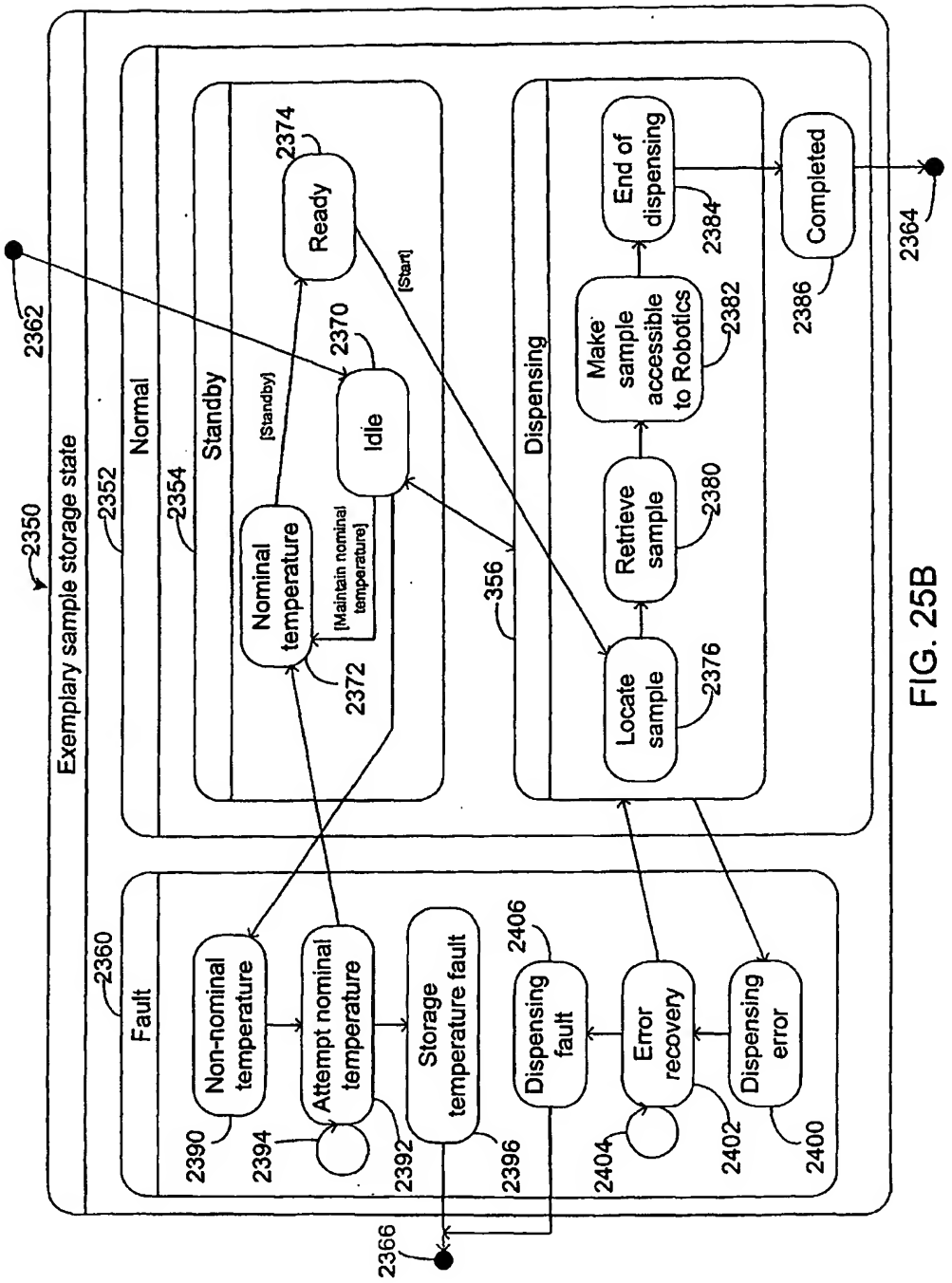

Exemplary sample storage state
2350
2362
Normal
2352
Standby
2354
Nominal temperature
[Maintain nominal temperature]
2372
[Standby]
Idle
2370
Ready
2374
[Start]
356
Dispensing
Locate sample
2376
Retrieve sample
2380
Make sample accessible to Robotics
2382
End of dispensing
2384
Completed
2386
2364
Fault
2360
Non-nominal temperature
2390
Attempt nominal temperature
2394
2392
Storage temperature fault
2396
Dispensing fault
2406
Error recovery
2404
Dispensing error
2402
2400
2366

FIG. 25B

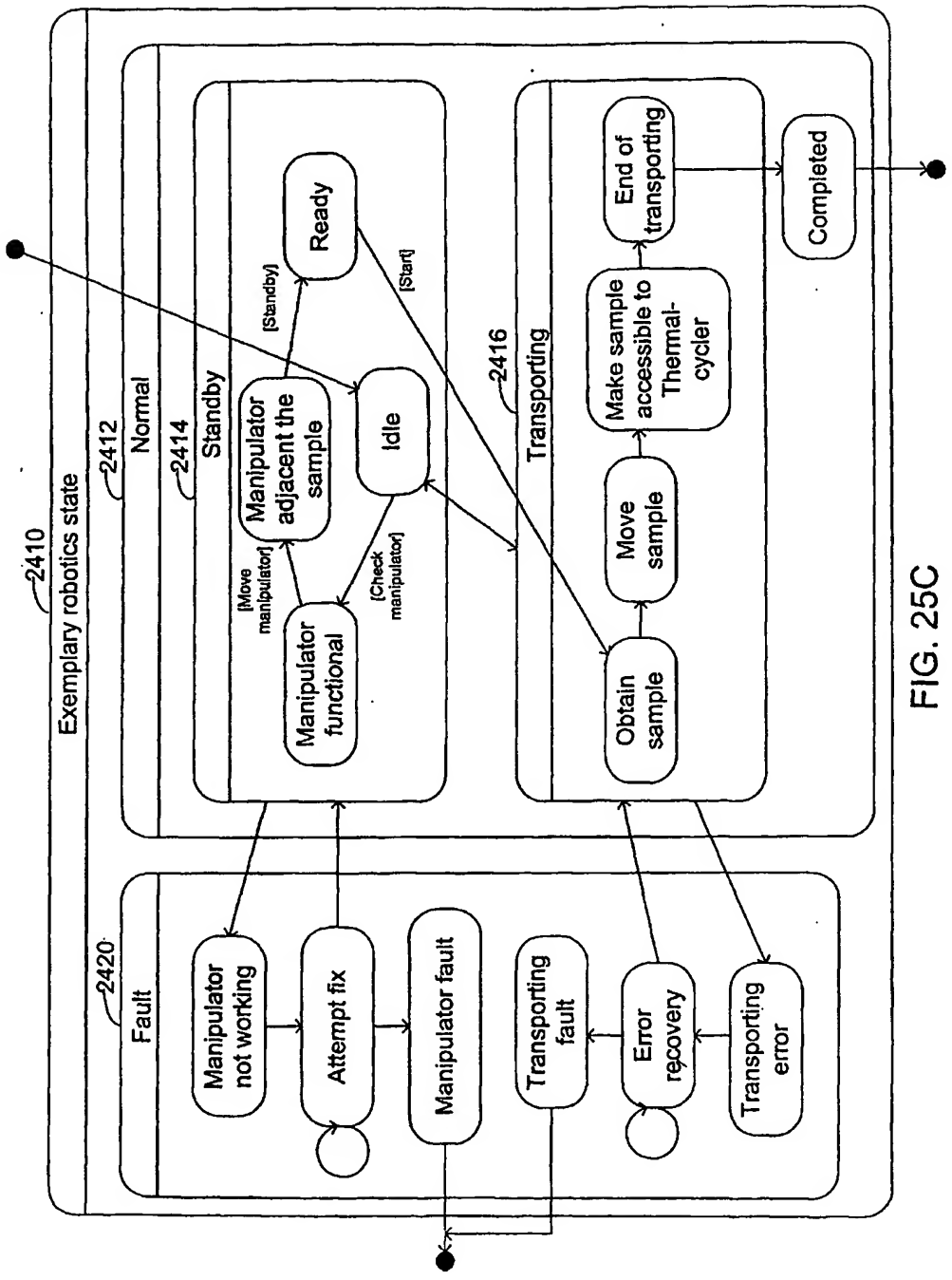

2410
Exemplary robotics state
2412
Normal
2414
Standby
Manipulator adjacent the sample
Idle
Ready
[Standby]
[Start]
[Move manipulator]
[Check manipulator]
Manipulator functional
2416
Transporting
Obtain sample
Move sample
Make sample accessible to Thermal-cycler
End of transporting
Completed
2420
Fault
Manipulator not working
Attempt fix
Manipulator fault
Transporting fault
Error recovery
Transporting error

FIG. 25C

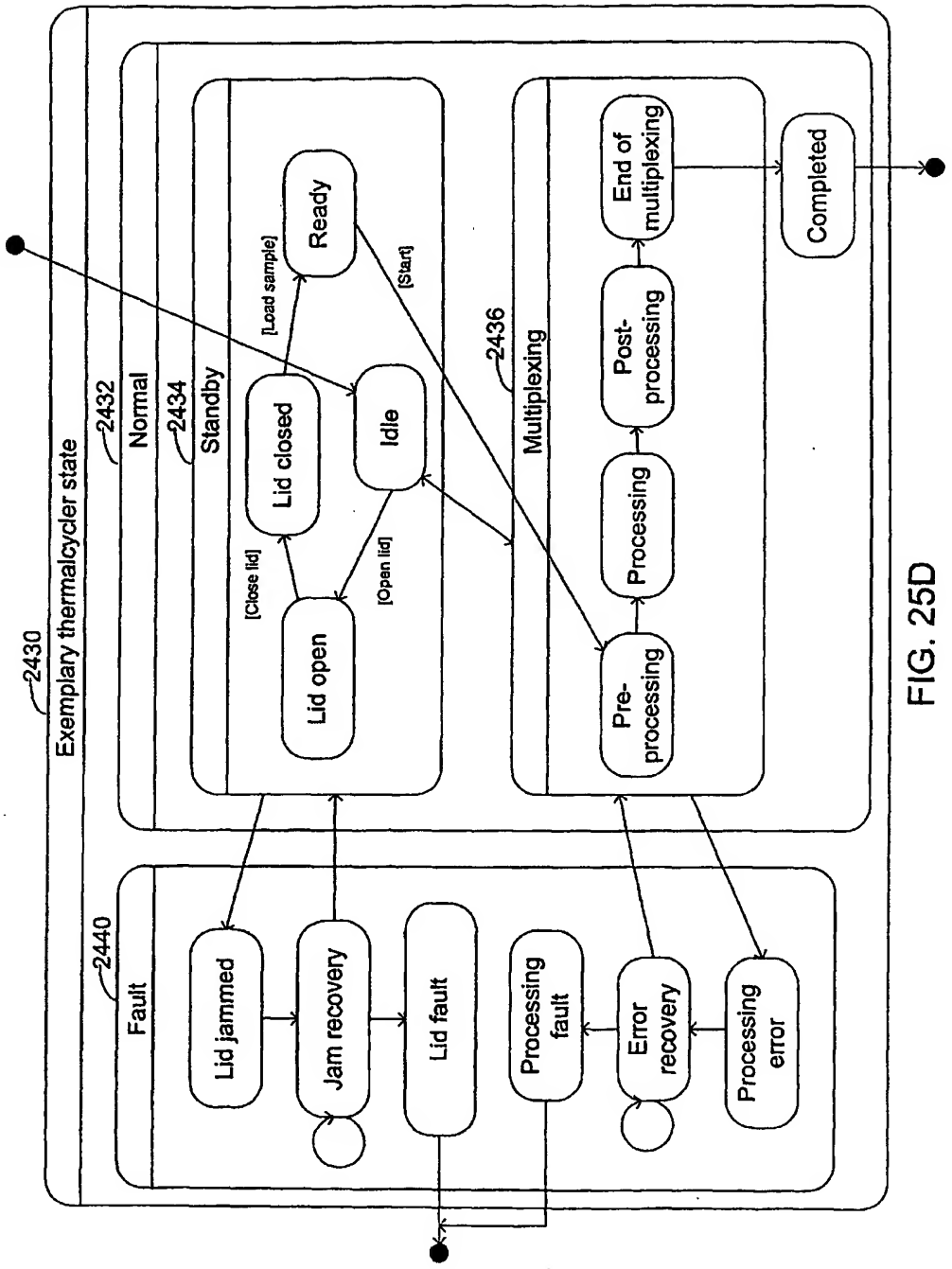

2430
Exemplary thermalcycler state
2432
Normal
2434
Standby
Lid closed
Ready
Idle
Lid open
[Load sample]
[Start]
[Close lid]
[Open lid]
2436
Multiplexing
Pre-processing
Processing
Post-processing
End of multiplexing
Completed
2440
Fault
Lid jammed
Jam recovery
Lid fault
Processing fault
Error recovery
Processing error

FIG. 25D

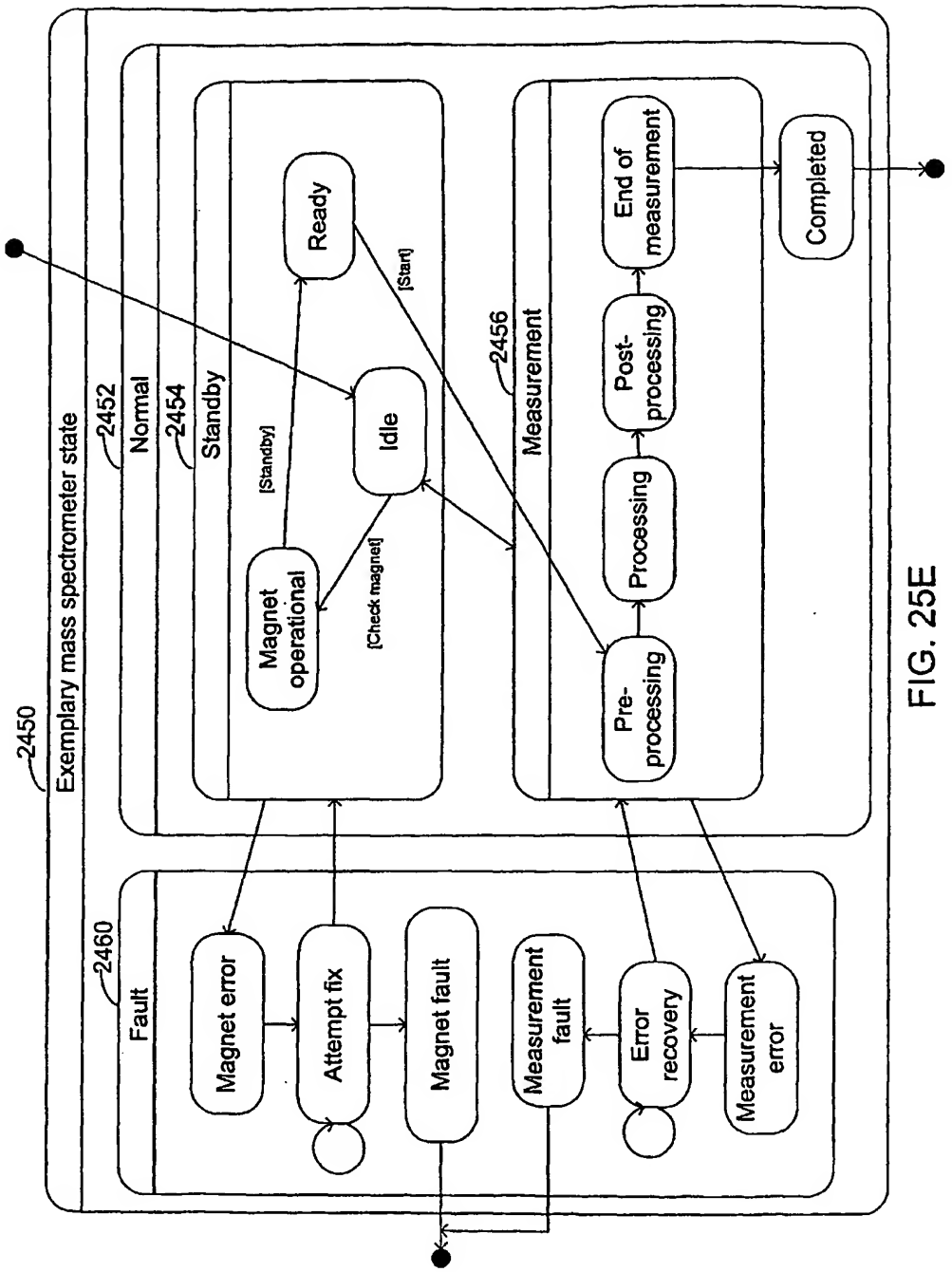

2450
Exemplary mass spectrometer state
2452
Normal
2454
Standby
Ready
[Start]
Idle
[Standby]
Magnet operational
[Check magnet]
2456
Measurement
Pre-processing
Processing
Post-processing
End of measurement
Completed
2460
Fault
Magnet error
Attempt fix
Magnet fault
Measurement fault
Error recovery
Measurement error

FIG. 25E

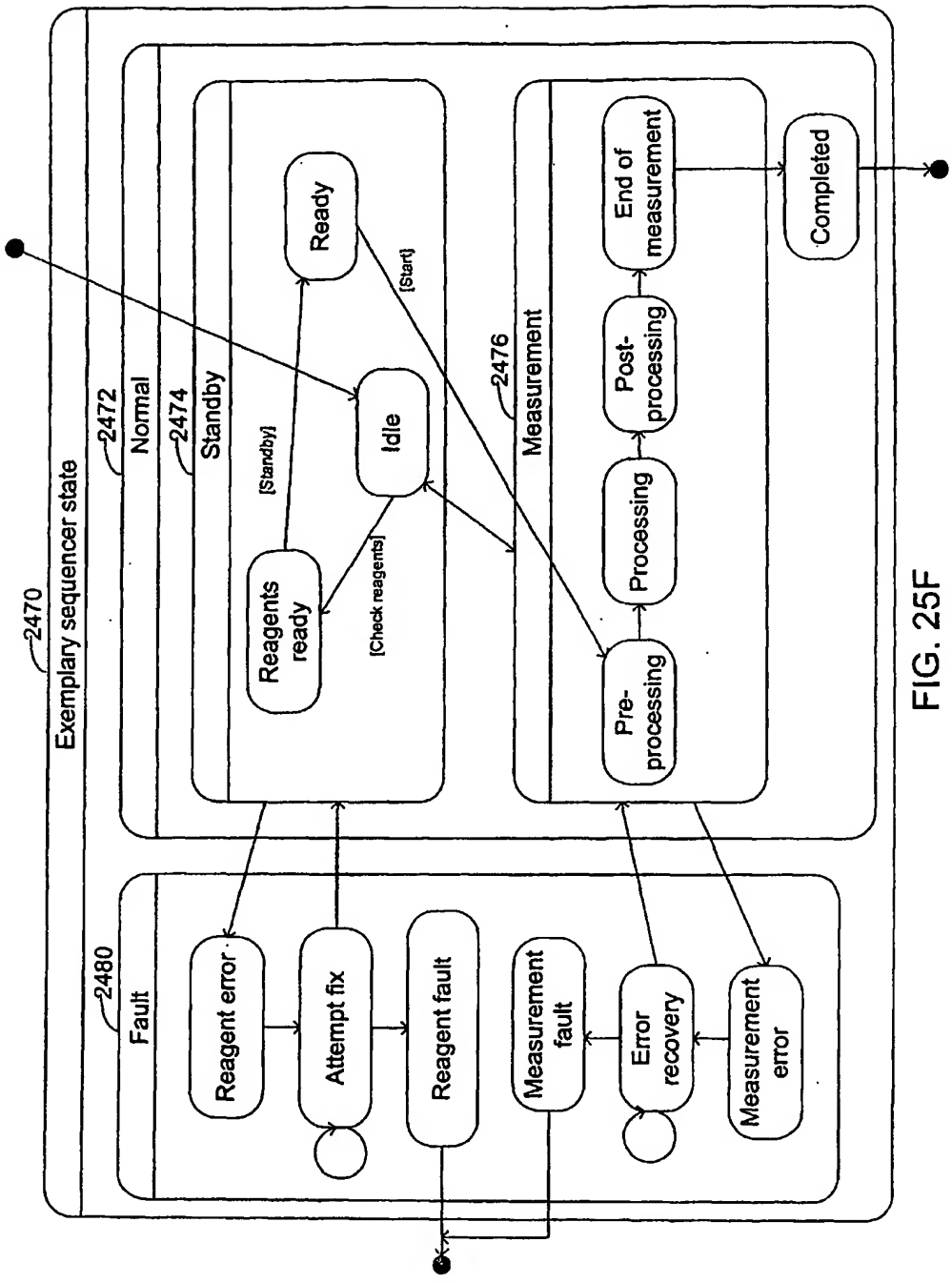

2470
Exemplary sequencer state
2472
Normal
2474
Standby
Ready
[Start]
[Standby]
Idle
Reagents ready
[Check reagents]
2476
Measurement
Pre-processing
Processing
Post-processing
End of measurement
Completed
2480
Fault
Reagent error
Attempt fix
Reagent fault
Measurement fault
Error recovery
Measurement error

FIG. 25F

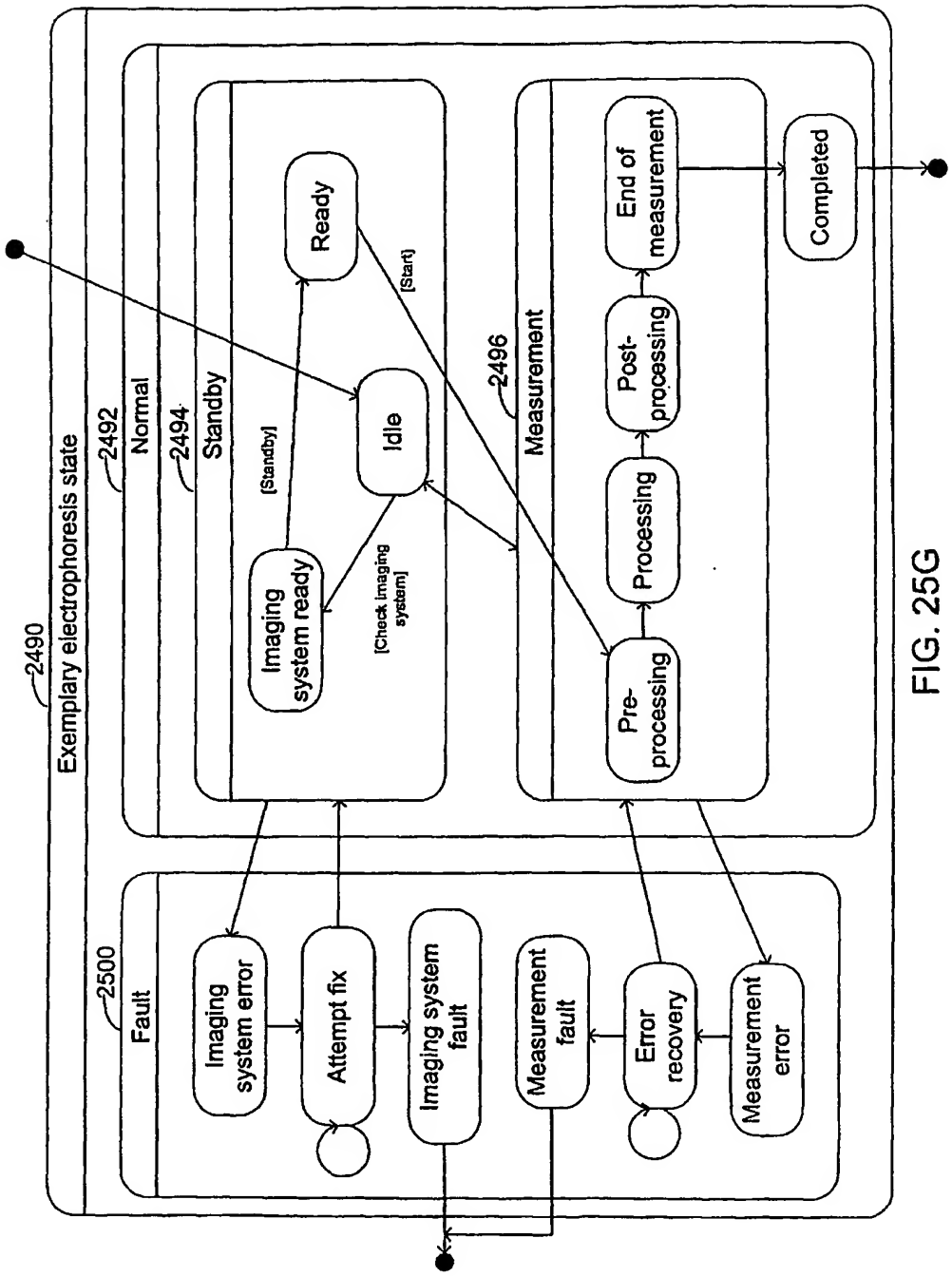

2490
Exemplary electrophoresis state
2492
Normal
2494
Standby
Ready
[Start]
Idle
[Standby]
Imaging system ready
[Check imaging system]
2496
Measurement
Pre-processing
Processing
Post-processing
End of measurement
Completed
2500
Fault
Imaging system error
Attempt fix
Imaging system fault
Measurement fault
Error recovery
Measurement error

FIG. 25G

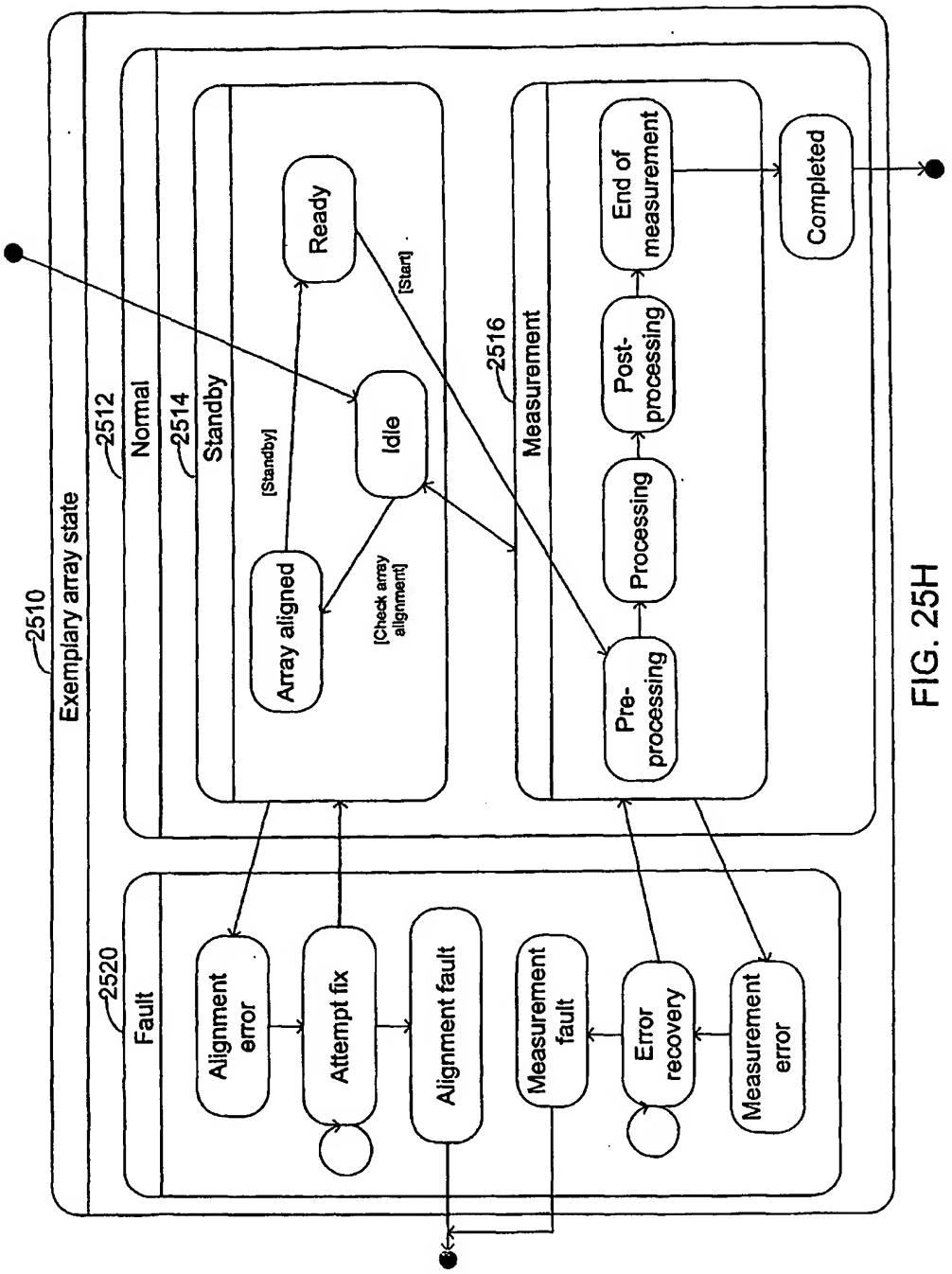

2510
Exemplary array state
2512
Normal
2514
Standby
Ready
[Start]
Idle
[Standby]
Array aligned
[Check array alignment]
2516
Measurement
End of measurement
Post-processing
Processing
Pre-processing
Completed
2520
Fault
Alignment error
Attempt fix
Alignment fault
Measurement fault
Error recovery
Measurement error

FIG. 25H

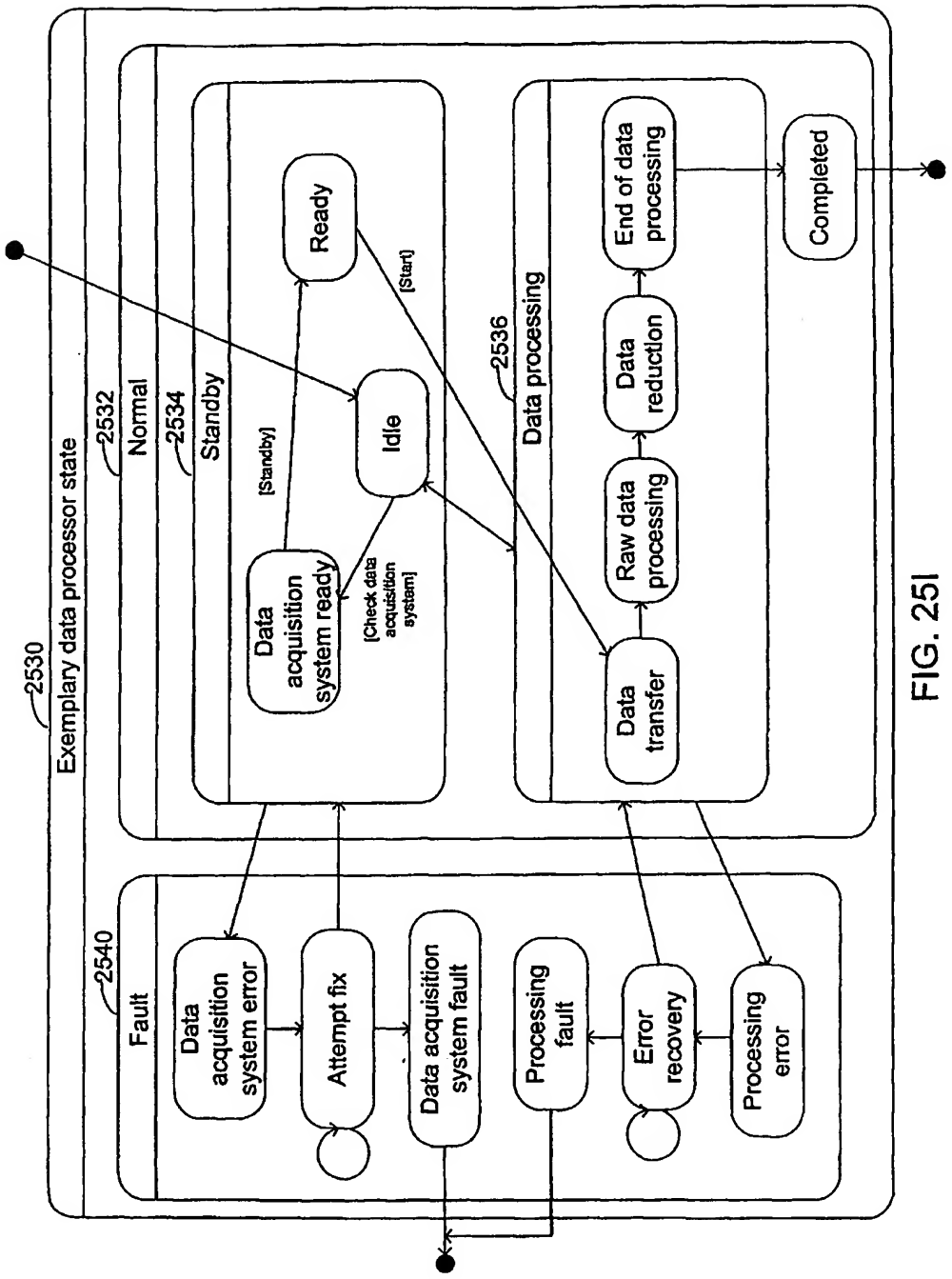

2530
Exemplary data processor state
2532
Normal
2534
Standby
Ready
[Start]
Idle
[Standby]
Data acquisition system ready
[Check data acquisition system]
2536
Data processing
End of data processing
Data reduction
Raw data processing
Data transfer
Completed
2540
Fault
Data acquisition system error
Attempt fix
Data acquisition system fault
Processing fault
Error recovery
Processing error

FIG. 25I

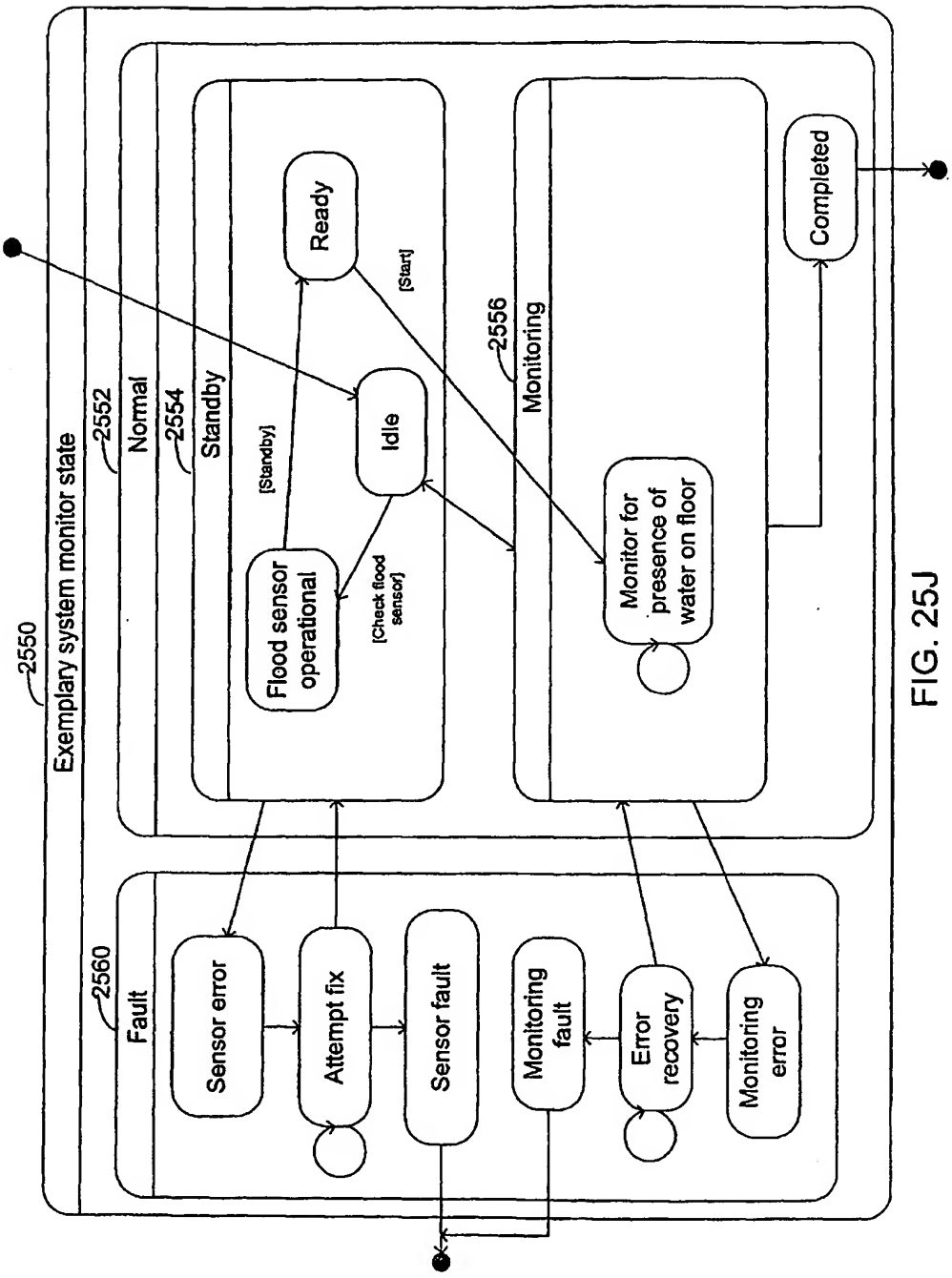

2550
Exemplary system monitor state
2552
Normal
2554
Standby
Ready
[Start]
Idle
[Standby]
Flood sensor operational
[Check flood sensor]
2556
Monitoring
Monitor for presence of water on floor
Completed
2560
Fault
Sensor error
Attempt fix
Sensor fault
Monitoring fault
Error recovery
Monitoring error

FIG. 25J

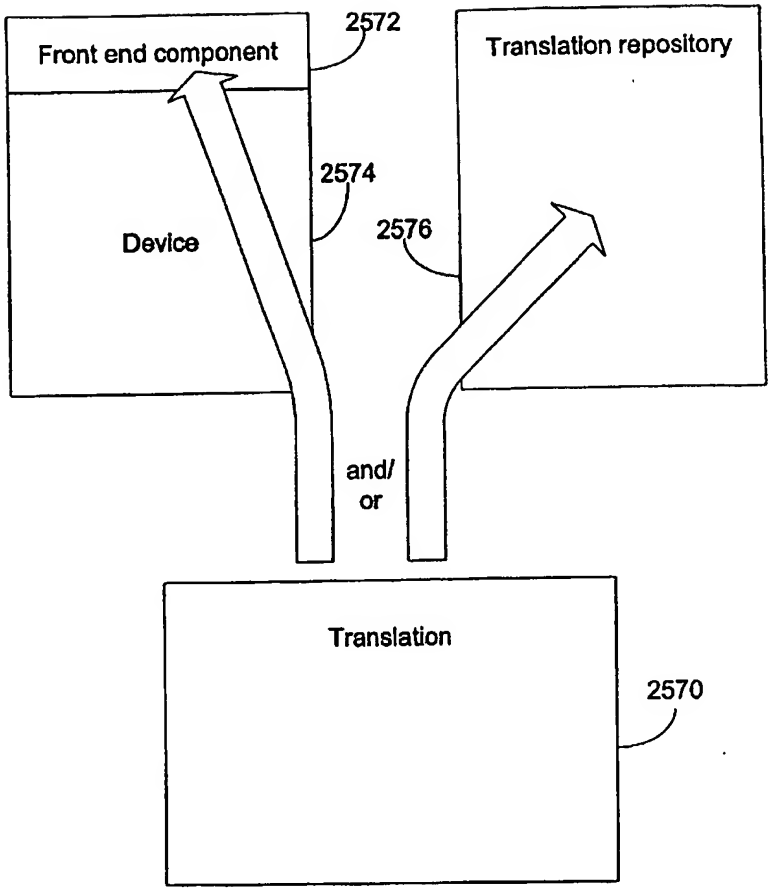

Front end component
2572
Translation repository
2574
Device
2576
and/
or
Translation
2570

FIG. 26A

2580

Exemplary translation for sample storage	
Device-specific states to STANDARDIZED STATES	STANDARDIZED COMMANDS to device-specific commands
If "idle," "nominal temperature," "ready," or "standby," then STANDBY. If "locate sample," "retrieve sample," "make sample accessible to robotics," end of dispensing," or "dispensing," then TASK. If "non-nominal temperature," "attempt nominal temperature," "storage temperature fault," "dispensing error," "error recovery," "dispensing fault," or "fault," then FAULT. If "completed," then COMPLETED.	If STANDBY command, - if device not in "standby" state, then go to "standby" state - if device already in "standby" state, then continue standby operation If TASK command, - if device not in "dispensing" state, then go to "dispensing" state - if device already in "dispensing" state, then continue dispensing operation If FAULT command, - if device not in "fault" state, then - if device in "dispensing" state and dispensing operation not affected by FAULT, then remain in "dispensing" state - if device in "dispensing" state and dispensing operation affected by FAULT, then go to "standby" state - if device in "standby" state, then remain in "standby" state - if device is already in "fault" state, then continue fault handling process If COMPLETED command, then remain "completed"

FIG. 26B

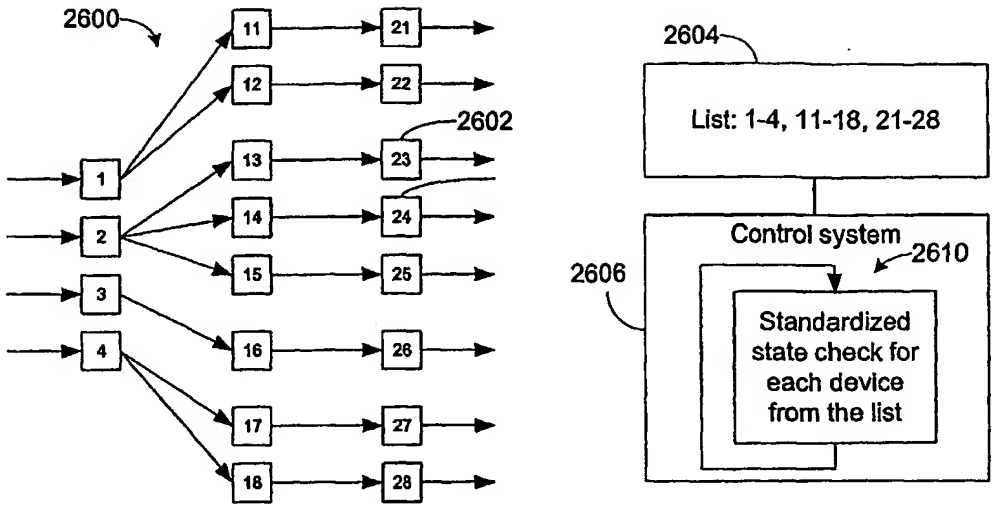


FIG. 27A

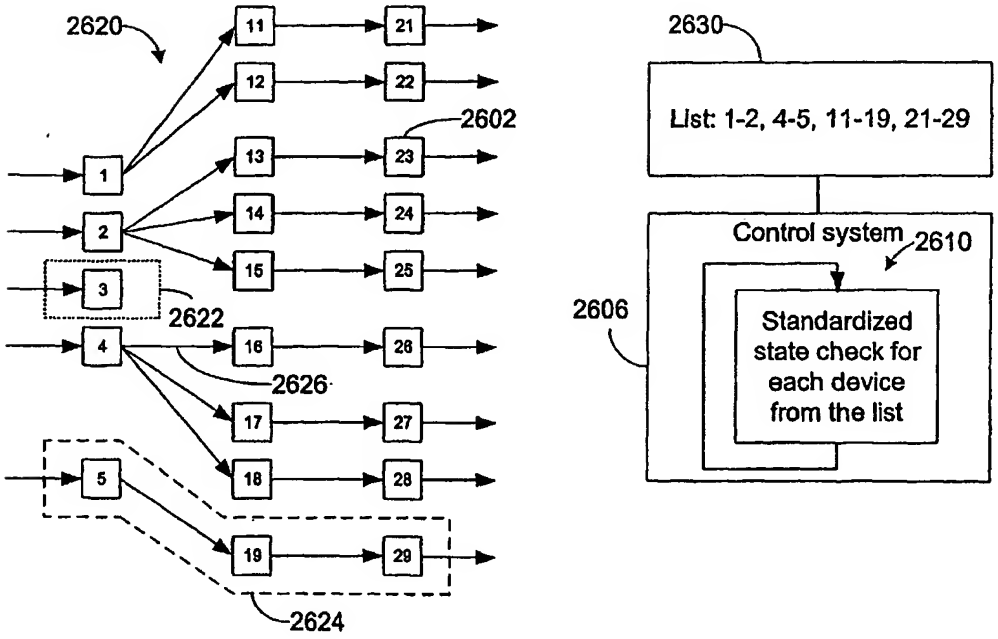


FIG. 27B

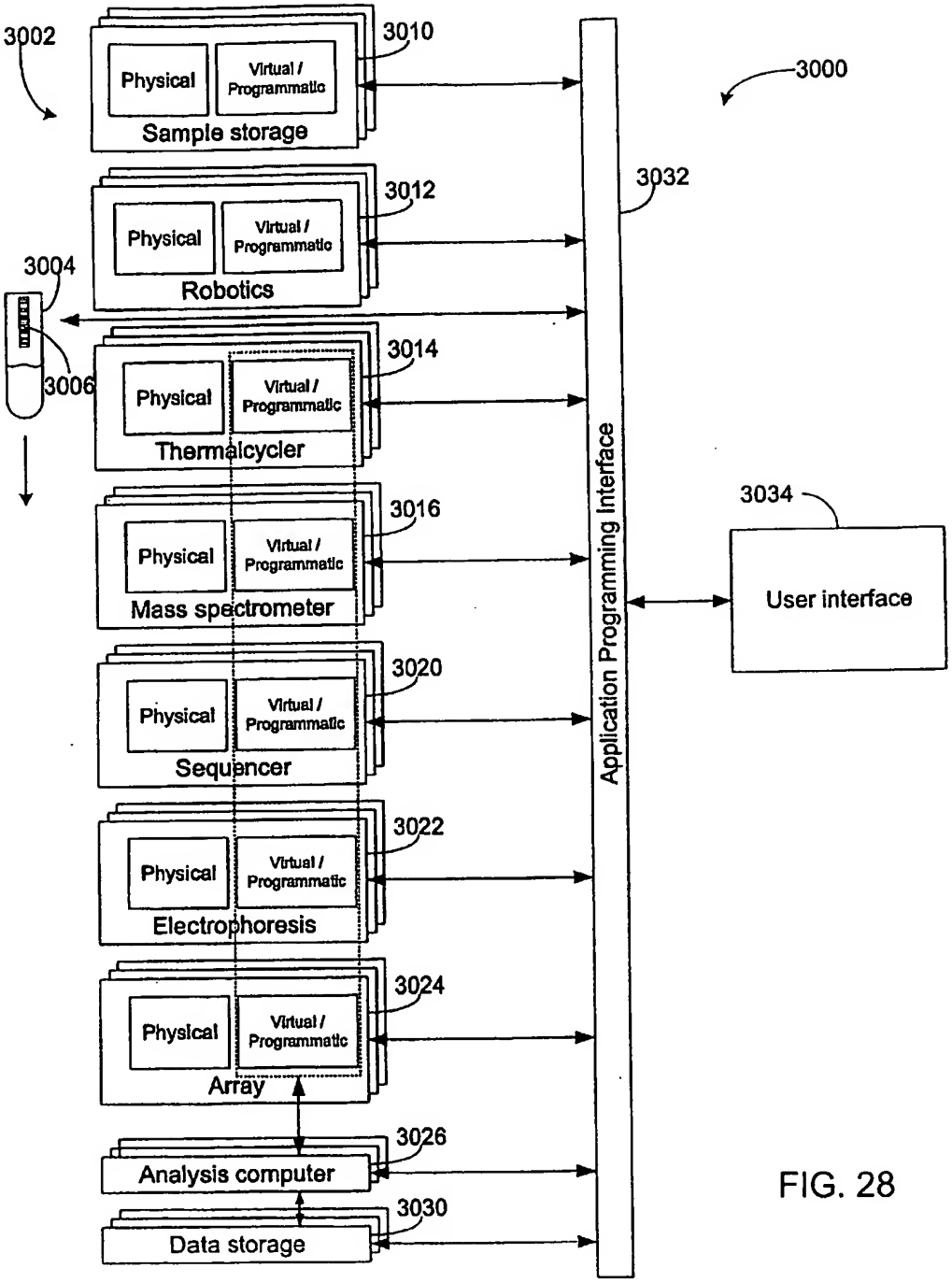

3002
3010
Physical
Virtual / Programmatic
Sample storage
3000
3032
3012
Physical
Virtual / Programmatic
Robotics
3004
3006
3014
Physical
Virtual / Programmatic
Thermalcycler
3016
Physical
Virtual / Programmatic
Mass spectrometer
3034
User interface
3020
Physical
Virtual / Programmatic
Sequencer
Application Programming Interface
3022
Physical
Virtual / Programmatic
Electrophoresis
3024
Physical
Virtual / Programmatic
Array
3026
Analysis computer
3030
Data storage

FIG. 28

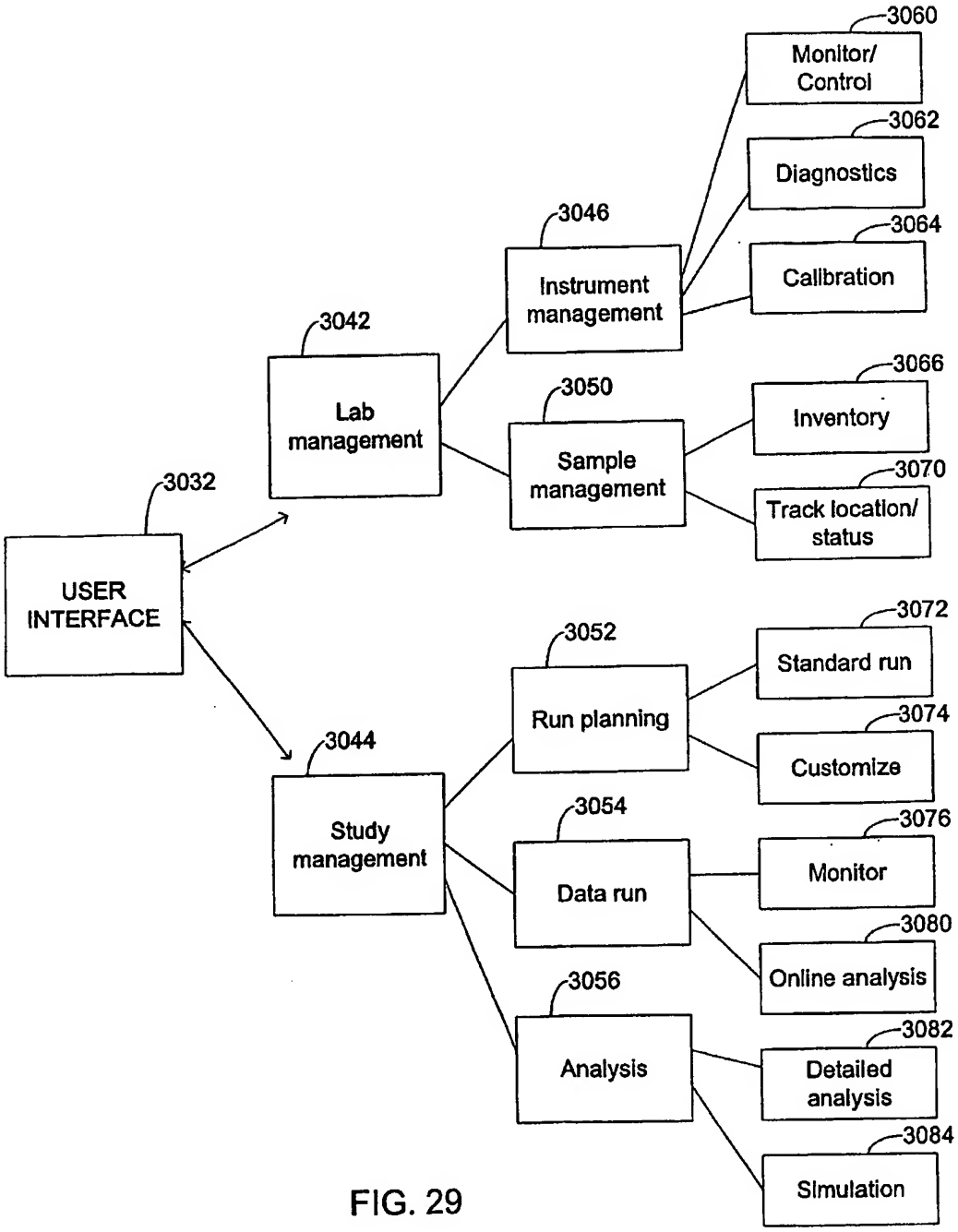

3060
Monitor/ Control
3062
Diagnostics
3064
Calibration
3046
Instrument management
3042
Lab management
3066
Inventory
3050
Sample management
3070
Track location/ status
3032
USER INTERFACE
3072
Standard run
3052
Run planning
3074
Customize
3044
Study management
3054
Data run
3076
Monitor
3080
Online analysis
3056
Analysis
3082
Detailed analysis
3084
Simulation

FIG. 29

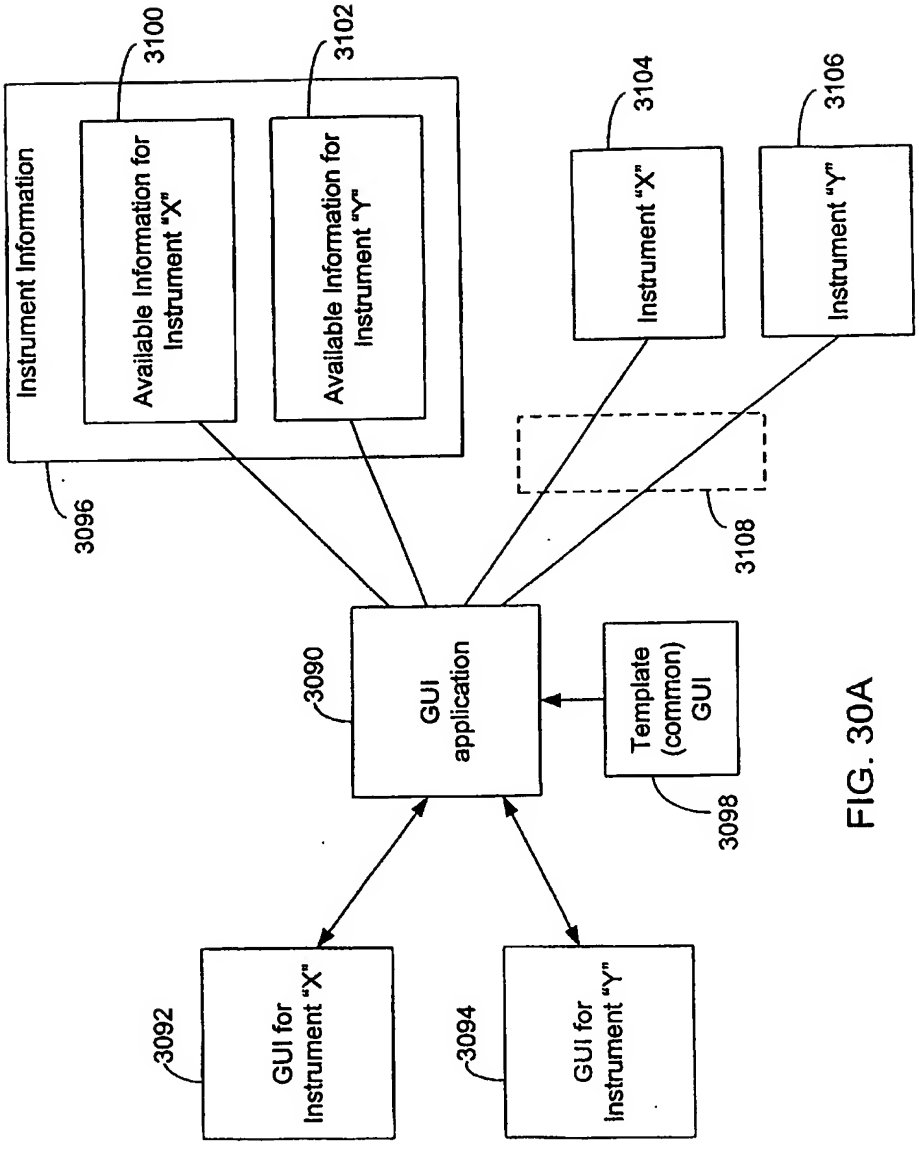

Instrument Information
3100
Available Information for Instrument "X"
3102
Available Information for Instrument "Y"
3096
3104
Instrument "X"
3106
Instrument "Y"
3108
3090
GUI application
Template (common) GUI
3098
3092
GUI for Instrument "X"
3094
GUI for Instrument "Y"

FIG. 30A

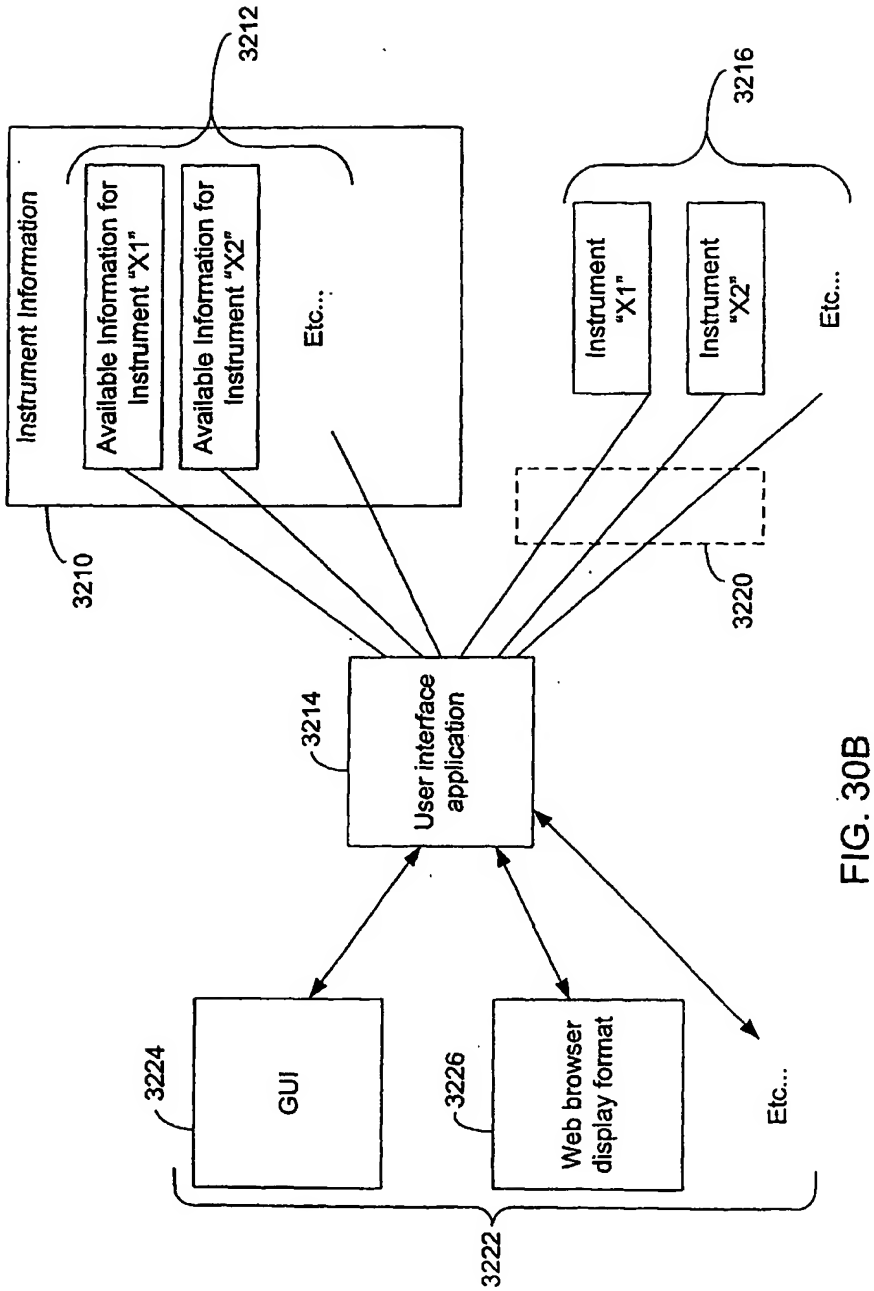

3212
Instrument Information
Available Information for Instrument "X1"
Available Information for Instrument "X2"
Etc...
3210
3216
Instrument "X1"
Instrument "X2"
Etc...
3220
3214
User interface application
3224
GUI
3226
Web browser display format
Etc...
3222

FIG. 30B

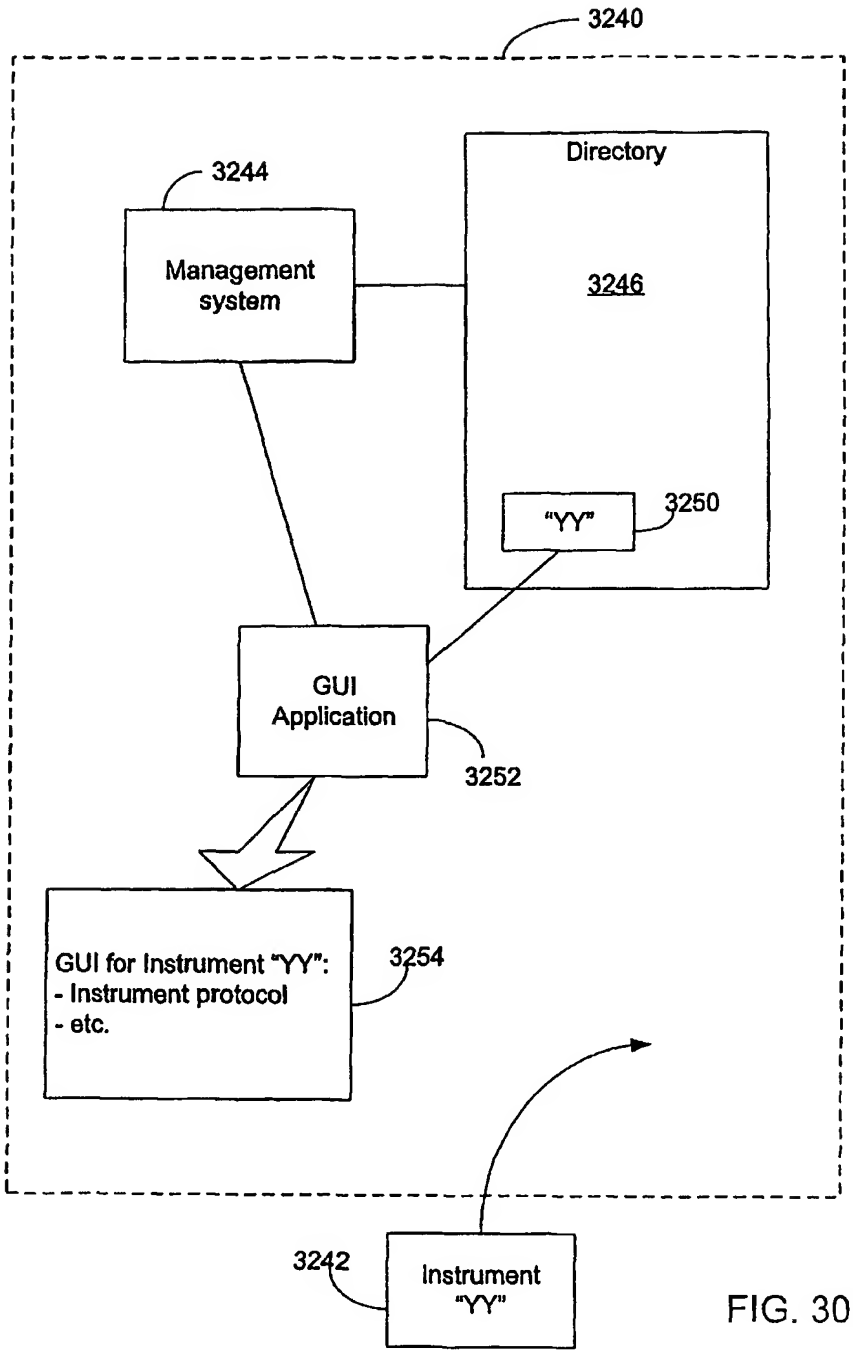

3240
Directory
3244
Management system
3246
"YY"
3250
GUI Application
3252
GUI for Instrument "YY":
- Instrument protocol
- etc.
3254
3242
Instrument "YY"

FIG. 30C

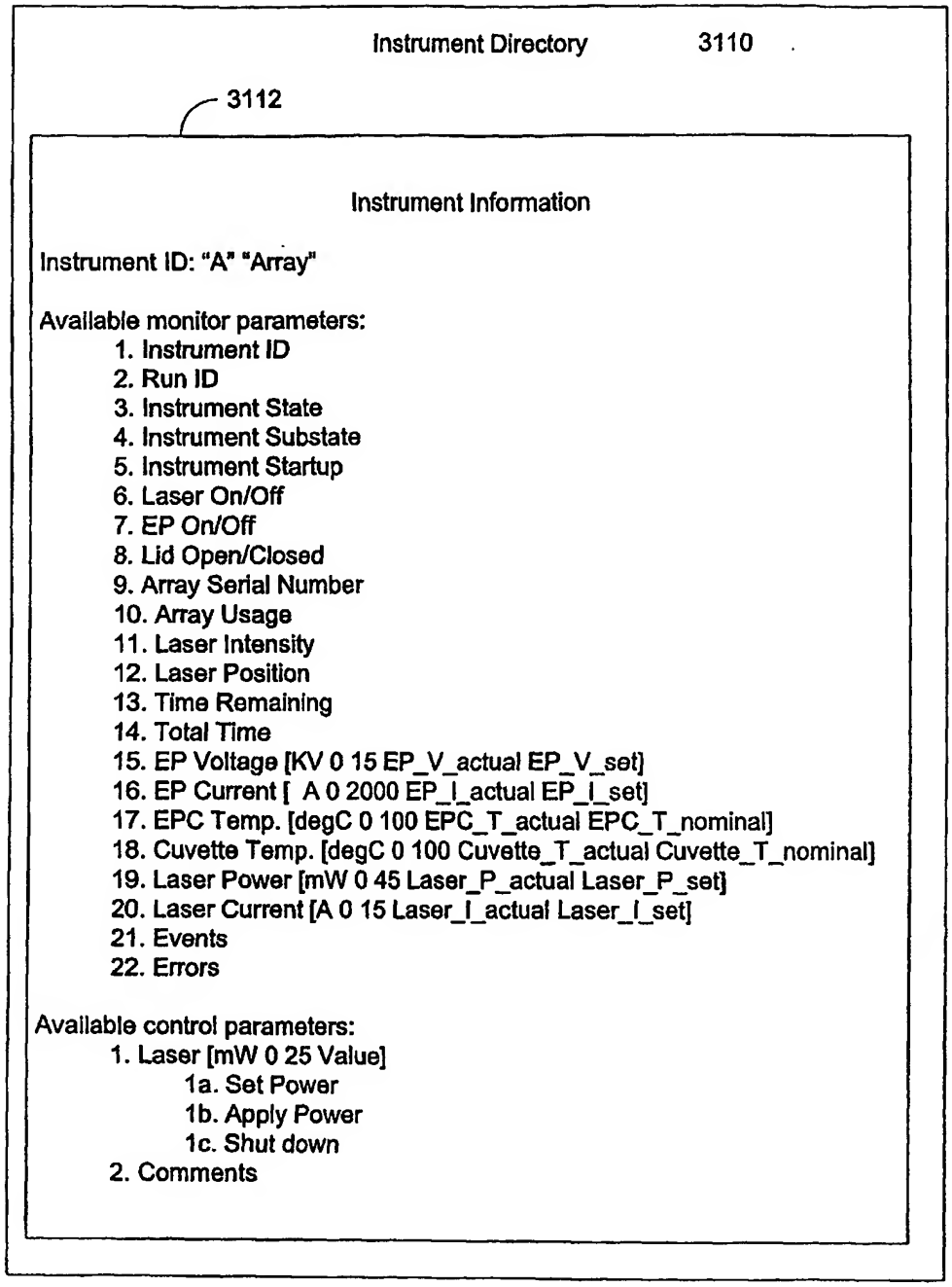

Instrument Directory 3110
3112
Instrument Information
Instrument ID: "A" "Array"
Available monitor parameters:
1. Instrument ID
2. Run ID
3. Instrument State
4. Instrument Substate
5. Instrument Startup
6. Laser On/Off
7. EP On/Off
8. Lid Open/Closed
9. Array Serial Number
10. Array Usage
11. Laser Intensity
12. Laser Position
13. Time Remaining
14. Total Time
15. EP Voltage [KV 0 15 EP_V_actual EP_V_set]
16. EP Current [A 0 2000 EP_I_actual EP_I_set]
17. EPC Temp. [degC 0 100 EPC_T_actual EPC_T_nominal]
18. Cuvette Temp. [degC 0 100 Cuvette_T_actual Cuvette_T_nominal]
19. Laser Power [mW 0 45 Laser_P_actual Laser_P_set]
20. Laser Current [A 0 15 Laser_I_actual Laser_I_set]
21. Events
22. Errors
Available control parameters:
1. Laser [mW 0 25 Value]
1a. Set Power
1b. Apply Power
1c. Shut down
2. Comments

FIG. 31A

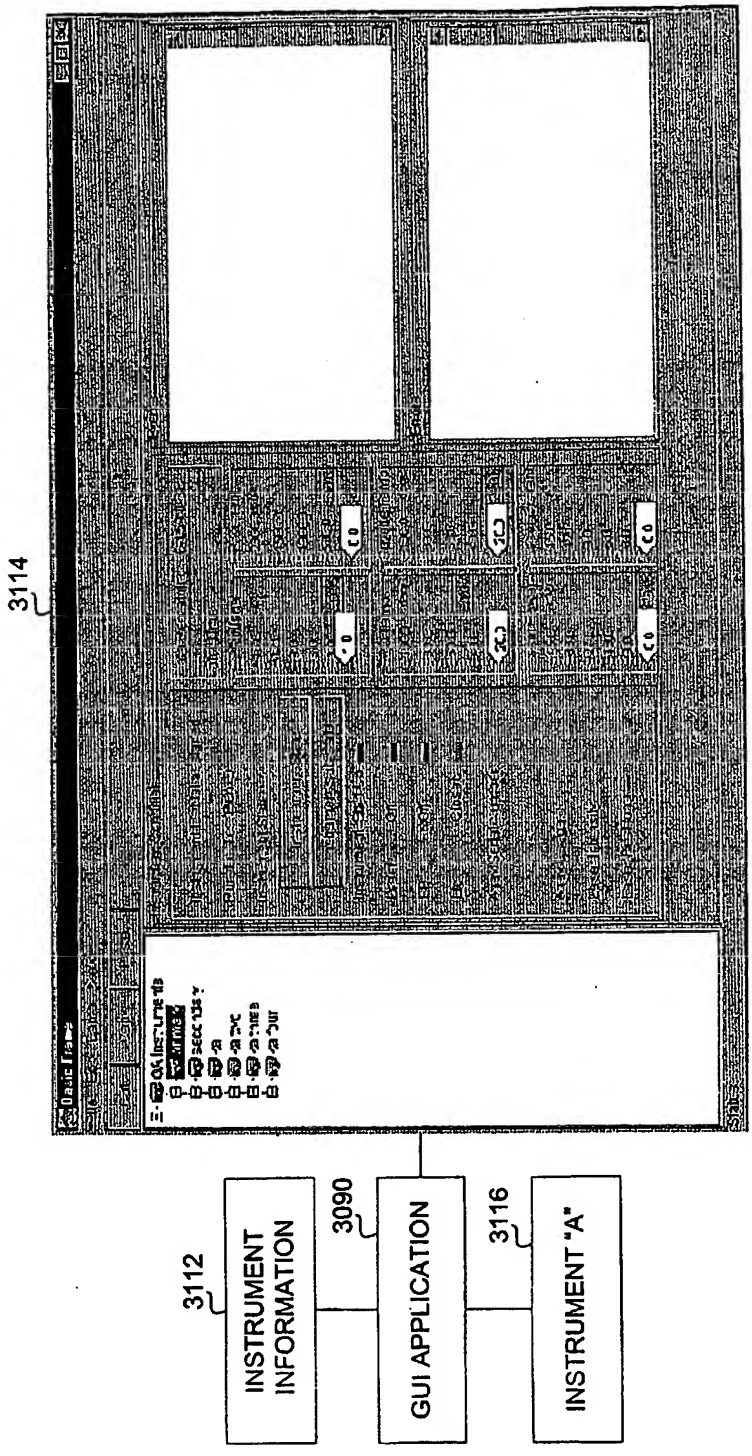

3114
3112
INSTRUMENT INFORMATION
3090
GUI APPLICATION
3116
INSTRUMENT "A"

FIGURE 31B

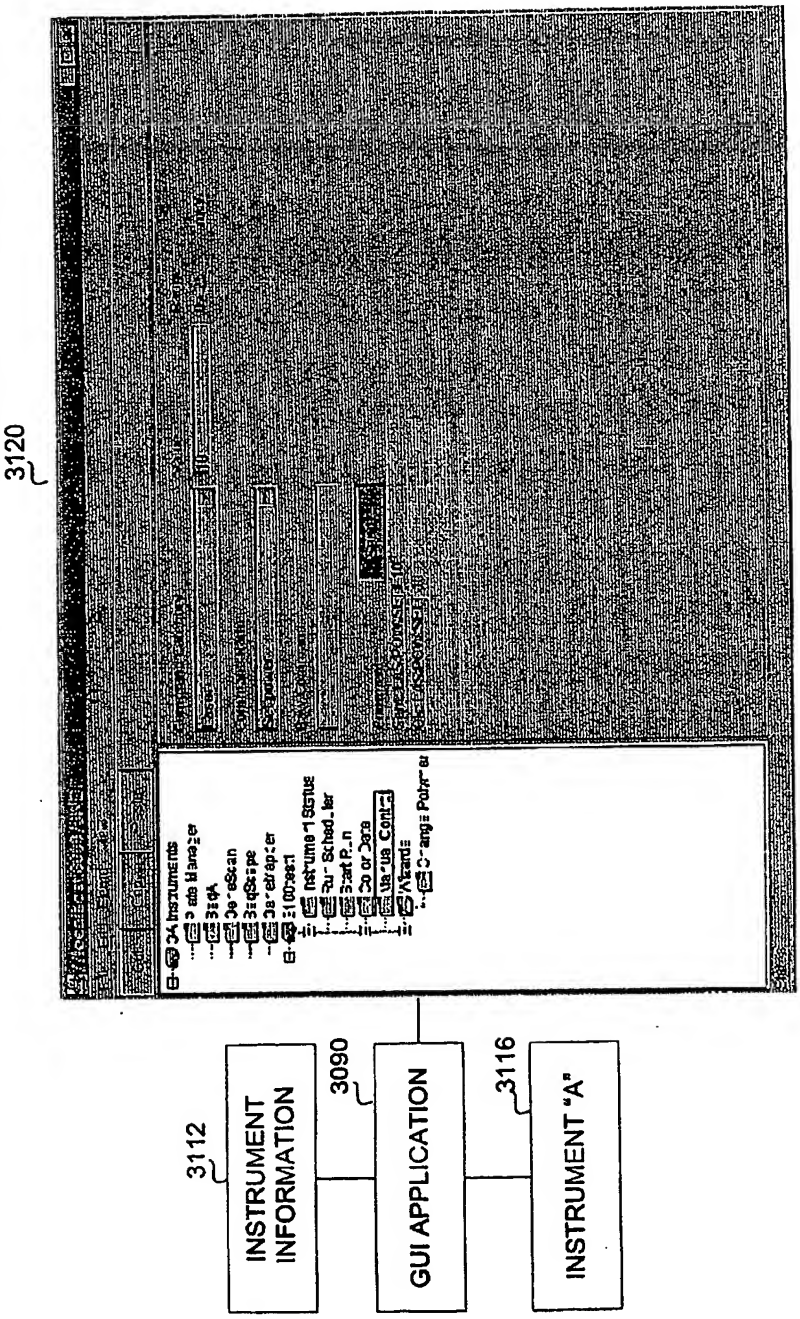
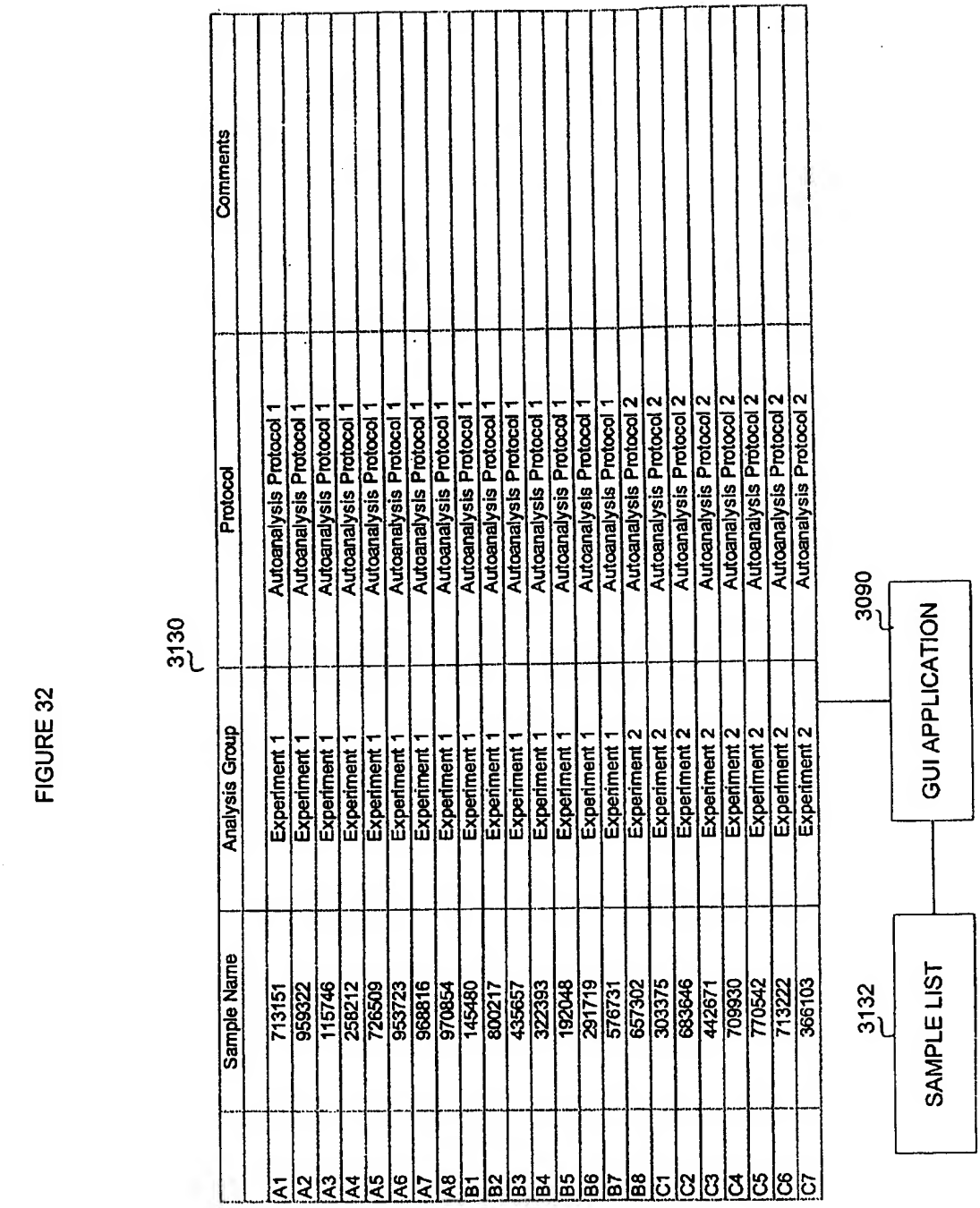

3120
3112
INSTRUMENT INFORMATION
3090
GUI APPLICATION
3116
INSTRUMENT "A"

FIGURE 31C

FIGURE 32


3130
Sample Name
Analysis Group
Protocol
Comments
A1 713151 Experiment 1 Autoanalysis Protocol 1
A2 959322 Experiment 1 Autoanalysis Protocol 1
A3 115746 Experiment 1 Autoanalysis Protocol 1
A4 258212 Experiment 1 Autoanalysis Protocol 1
A5 726509 Experiment 1 Autoanalysis Protocol 1
A6 953723 Experiment 1 Autoanalysis Protocol 1
A7 968816 Experiment 1 Autoanalysis Protocol 1
A8 970854 Experiment 1 Autoanalysis Protocol 1
B1 145480 Experiment 1 Autoanalysis Protocol 1
B2 800217 Experiment 1 Autoanalysis Protocol 1
B3 435657 Experiment 1 Autoanalysis Protocol 1
B4 322393 Experiment 1 Autoanalysis Protocol 1
B5 192048 Experiment 1 Autoanalysis Protocol 1
B6 291719 Experiment 1 Autoanalysis Protocol 1
B7 576731 Experiment 1 Autoanalysis Protocol 1
B8 657302 Experiment 2 Autoanalysis Protocol 2
C1 303375 Experiment 2 Autoanalysis Protocol 2
C2 683646 Experiment 2 Autoanalysis Protocol 2
C3 442671 Experiment 2 Autoanalysis Protocol 2
C4 709930 Experiment 2 Autoanalysis Protocol 2
C5 770542 Experiment 2 Autoanalysis Protocol 2
C6 713222 Experiment 2 Autoanalysis Protocol 2
C7 366103 Experiment 2 Autoanalysis Protocol 2
3090
GUI APPLICATION
3132
SAMPLE LIST

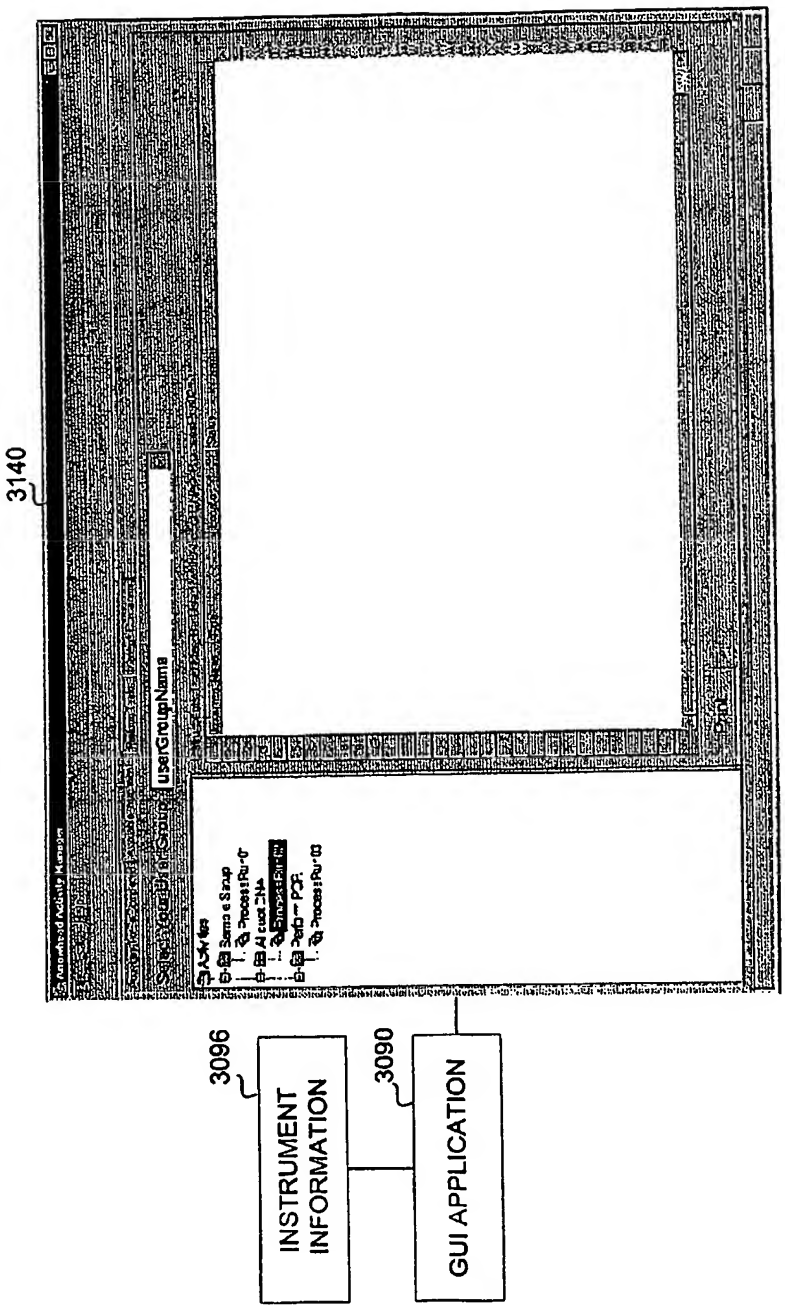


FIGURE 33A

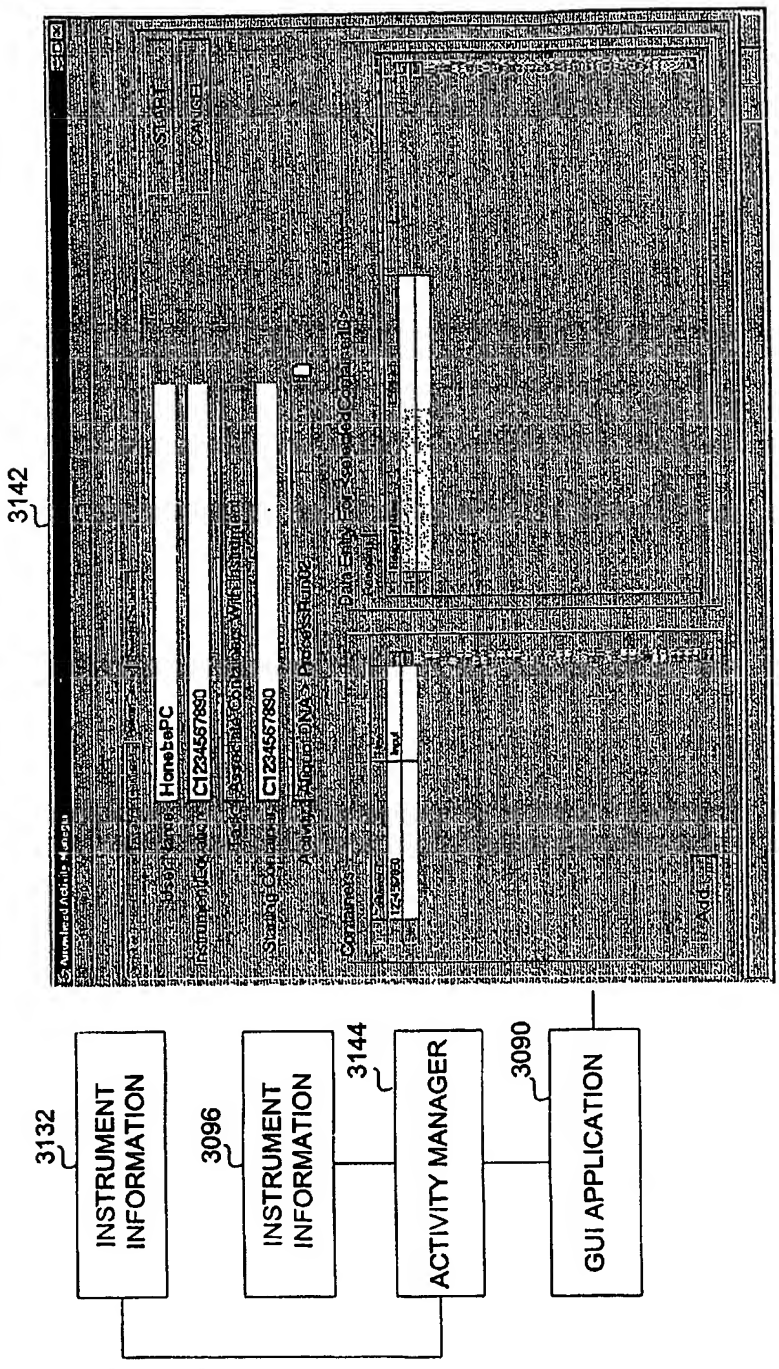

3142
3132
INSTRUMENT INFORMATION
3096
INSTRUMENT INFORMATION
3144
ACTIVITY MANAGER
3090
GUI APPLICATION

FIGURE 33B

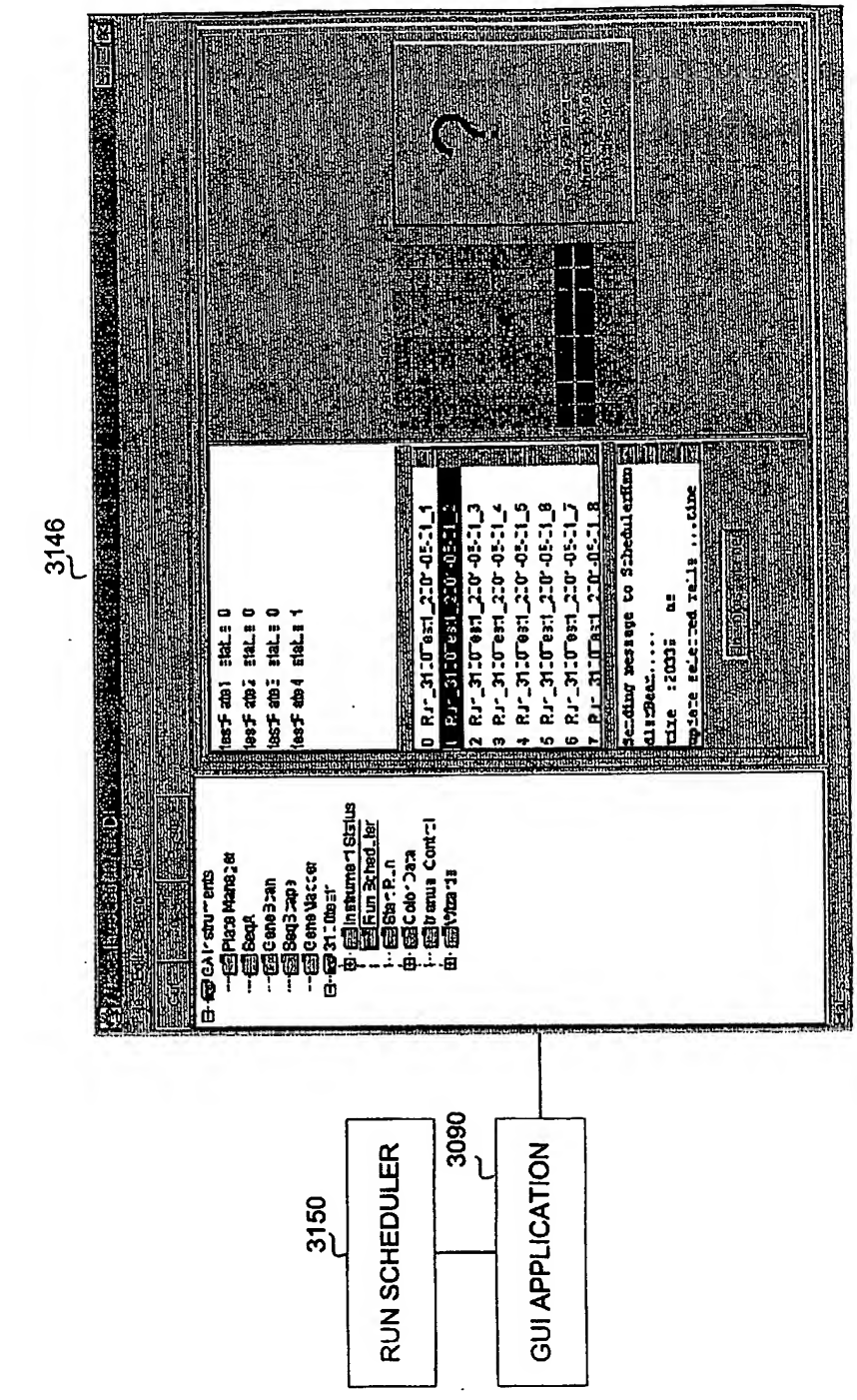


FIGURE 33C

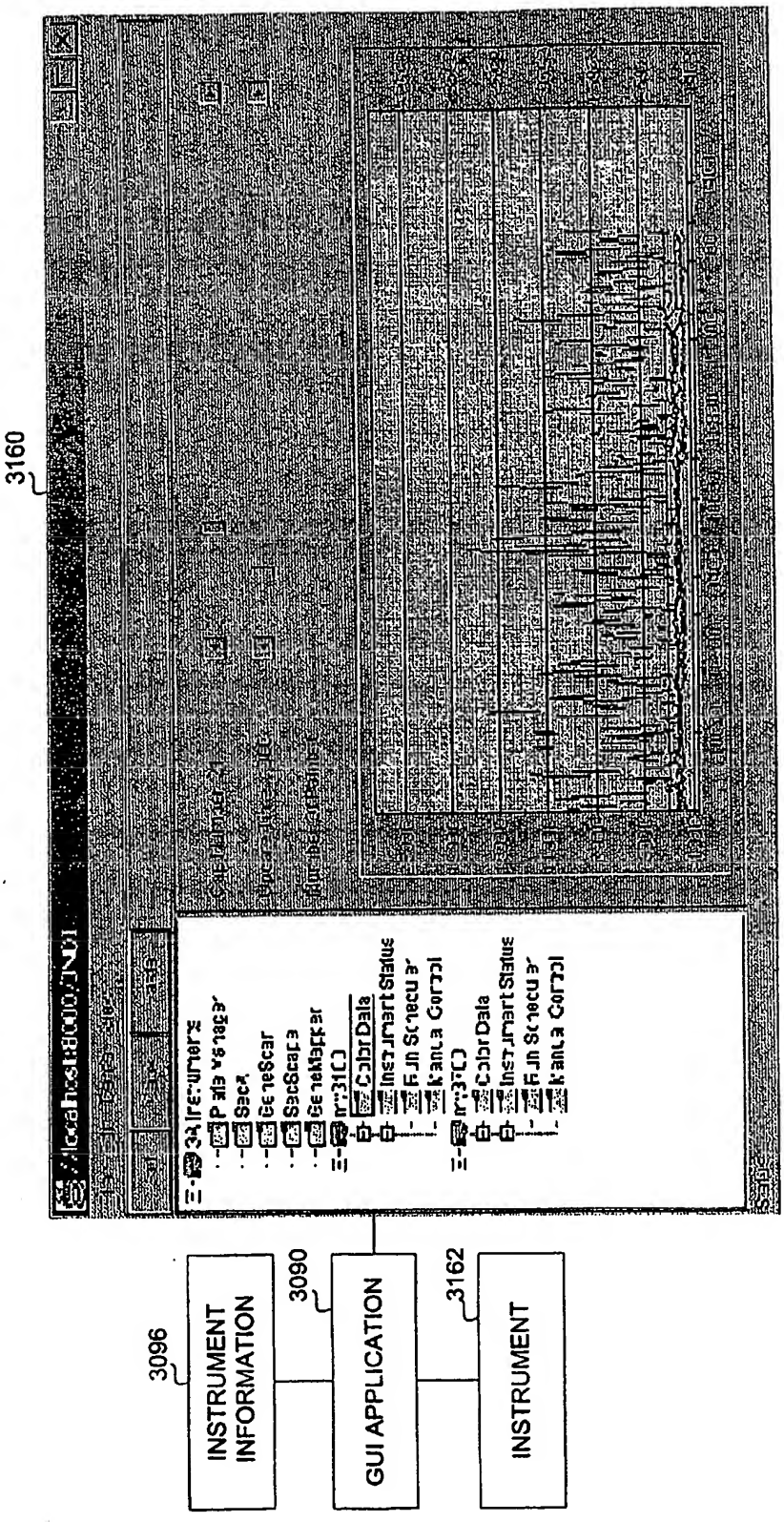

3160
3096
INSTRUMENT INFORMATION
3090
GUI APPLICATION
3162
INSTRUMENT

FIGURE 34A

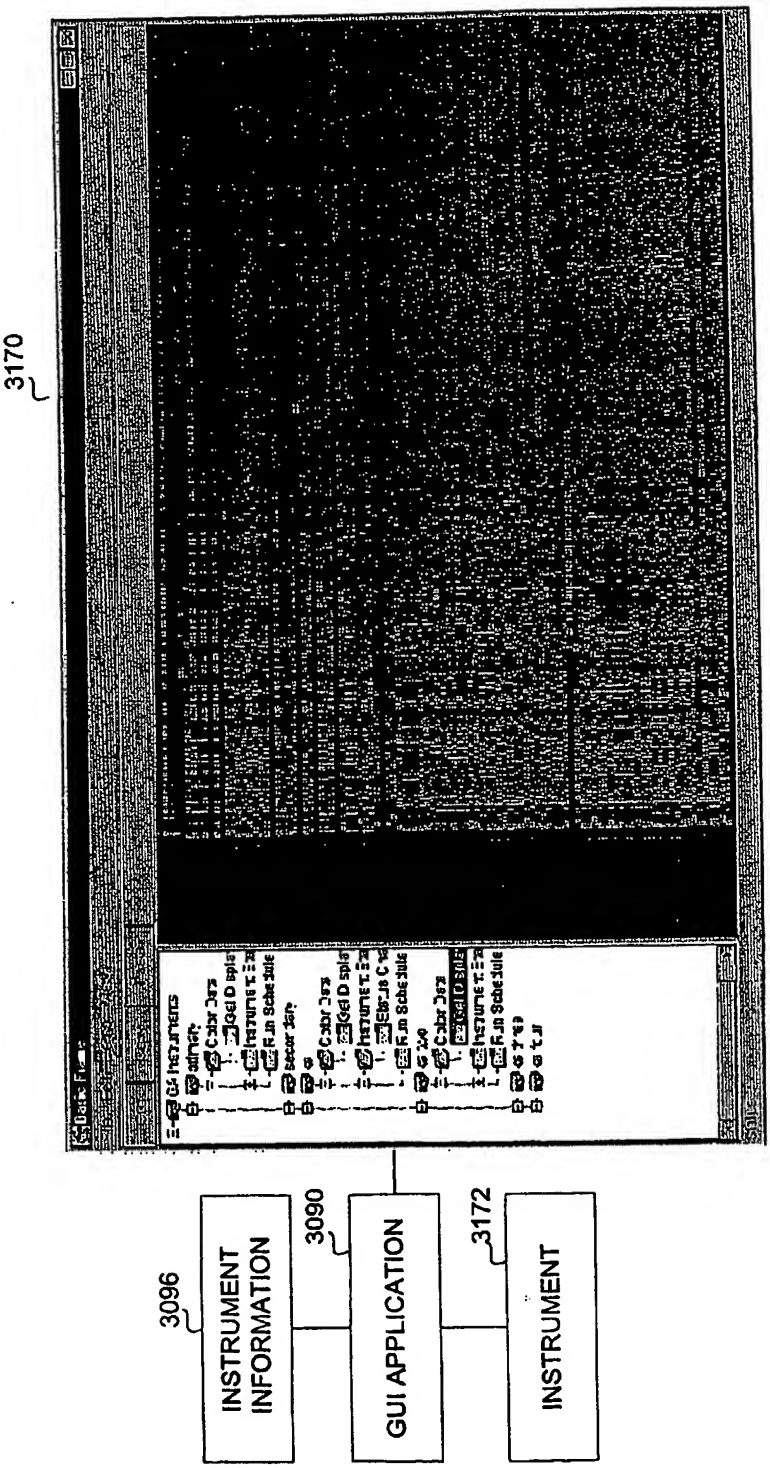

3170
3096
INSTRUMENT INFORMATION
3090
GUI APPLICATION
3172
INSTRUMENT

FIGURE 34B

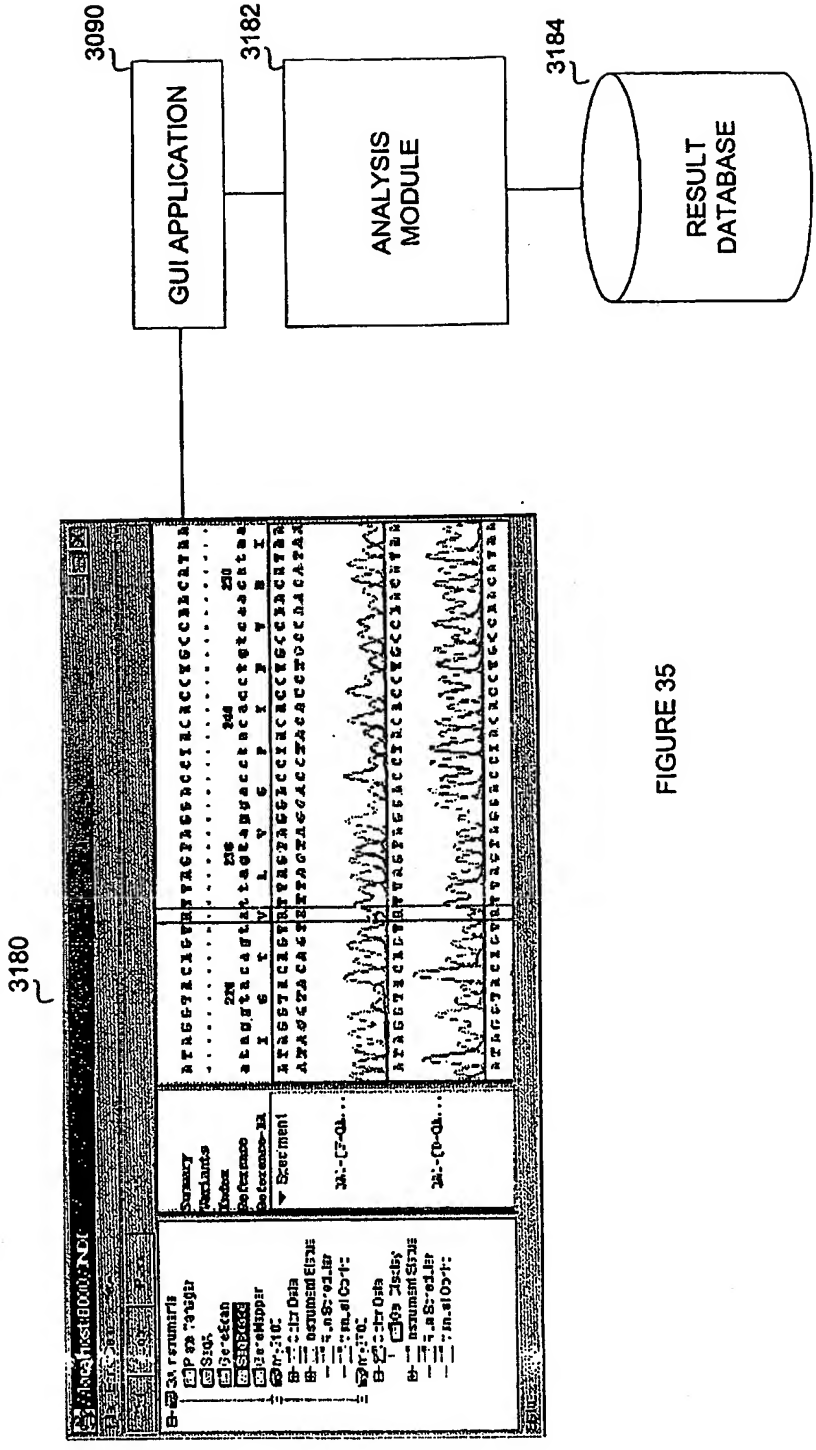

3090
GUI APPLICATION
3182
ANALYSIS MODULE
3184
RESULT DATABASE
3180

FIGURE 35

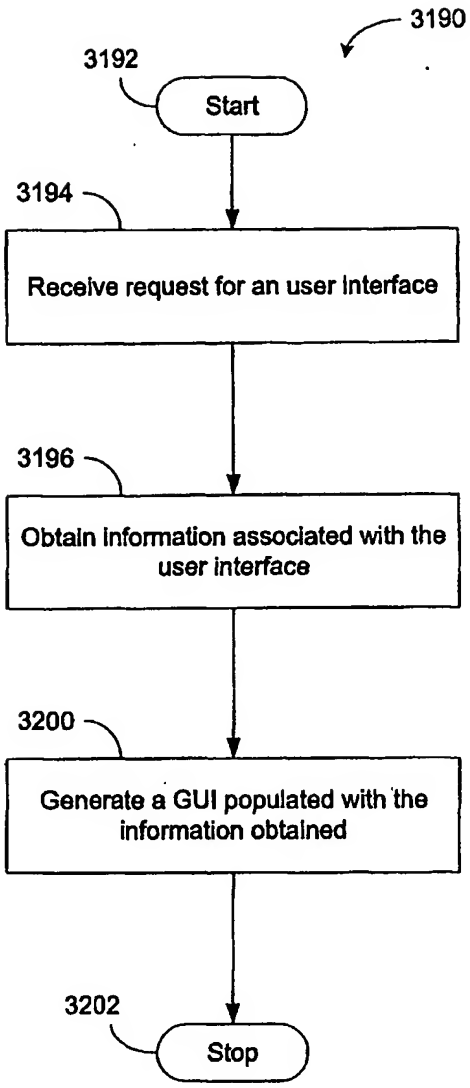

3190
3192
Start
3194
Receive request for an user interface
3196
Obtain information associated with the user interface
3200
Generate a GUI populated with the information obtained
3202
Stop

FIG. 36

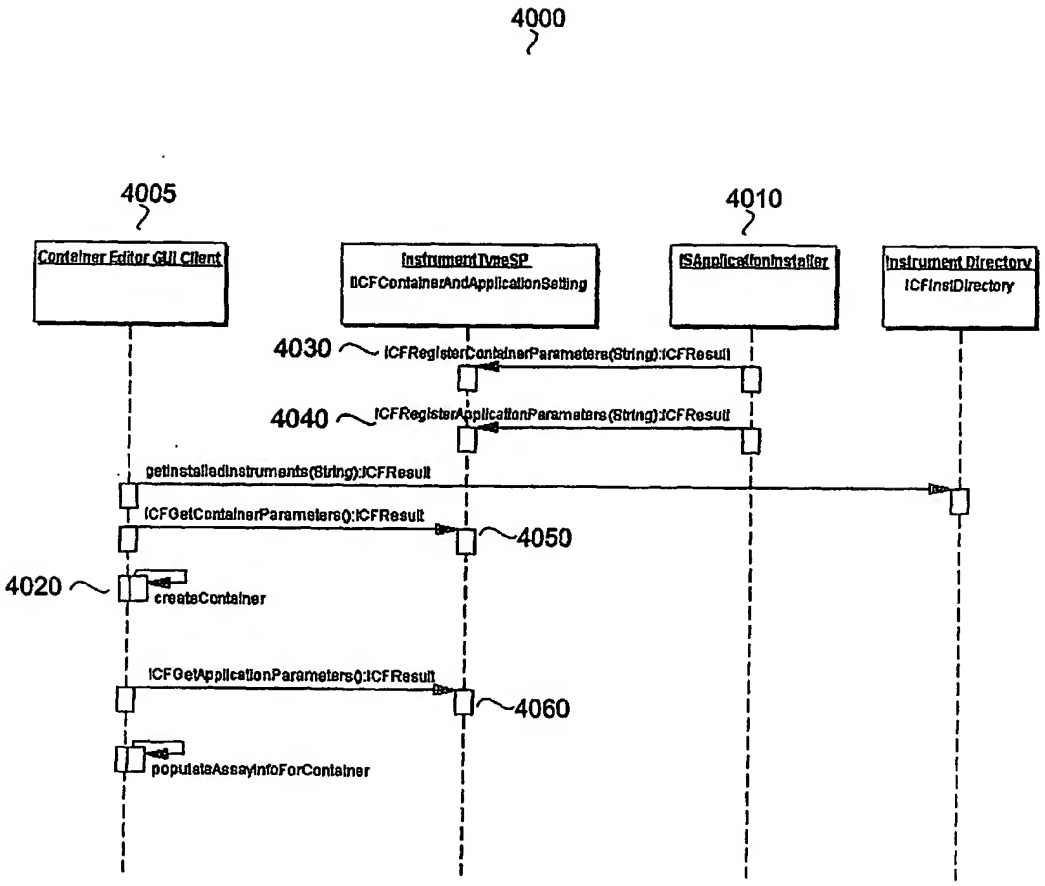

4000
4005
4010
Container Editor GUI Client
InstrumentTypeSP
IICFContainerAndApplicationSetting
ISApplicationInstaller
Instrument Directory
ICFInstDirectory
4030
ICFRegisterContainerParameters(String):ICFResult
4040
ICFRegisterApplicationParameters(String):ICFResult
getInstalledInstruments(String):ICFResult
ICFGetContainerParameters():ICFResult
4050
4020
createContainer
ICFGetApplicationParameters():ICFResult
4060
populateAssayInfoForContainer

Fig. 37

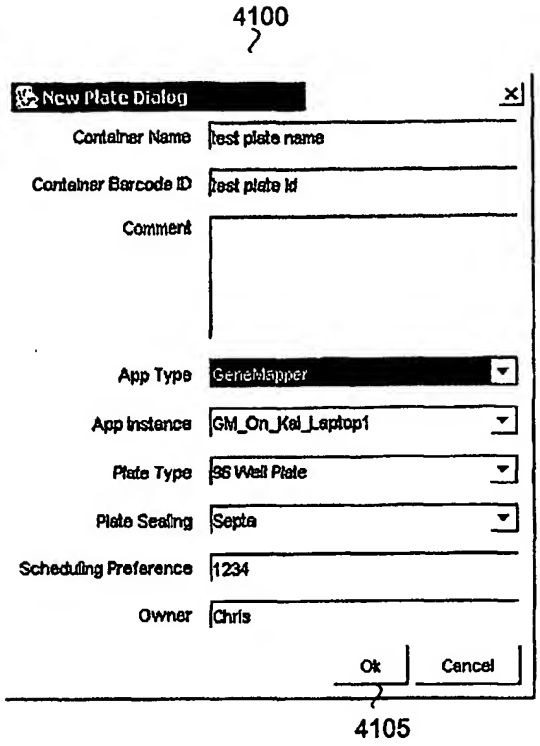

4100
New Plate Dialog
Container Name
test plate name
Container Barcode ID
test plate id
Comment
App Type
GeneMapper
App Instance
GM_On_Kai_Laptop1
Plate Type
96 Well Plate
Plate Sealing
Septa
Scheduling Preference
1234
Owner
Chris
Ok
Cancel
4105

Fig. 38

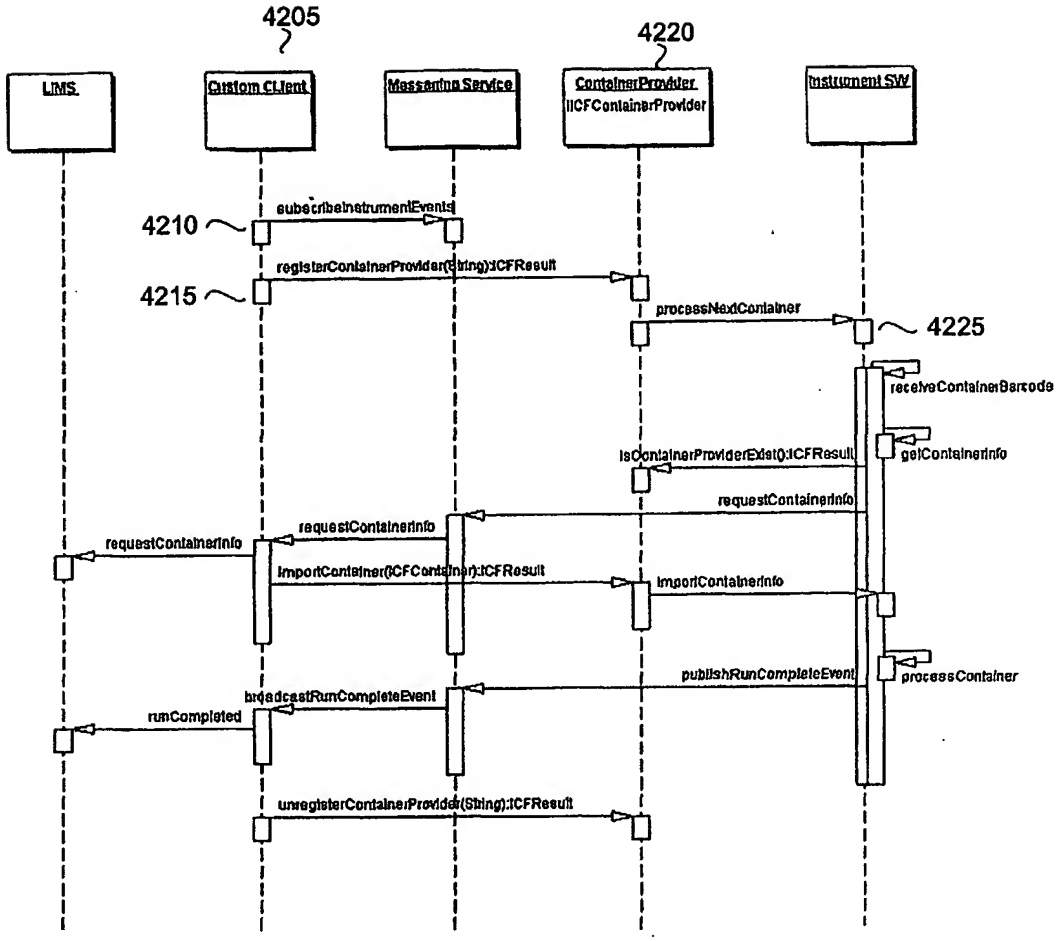

4205
4220
LIMS
Custom CLient
Messaging Service
ContainerProvider
IICFContainerProvider
Instrument SW
4210
subscribeInstrumentEvents()
4215
registerContainerProvider(String):ICFResult
processNextContainer
4225
receiveContainerBarcode
getContainerInfo
isContainerProviderExist():ICFResult
requestContainerInfo
requestContainerInfo
requestContainerInfo
importContainer(ICFContainer):ICFResult
importContainerInfo
processContainer
publishRunCompleteEvent
broadcastRunCompleteEvent
runCompleted
unregisterContainerProvider(String):ICFResult

Fig. 39

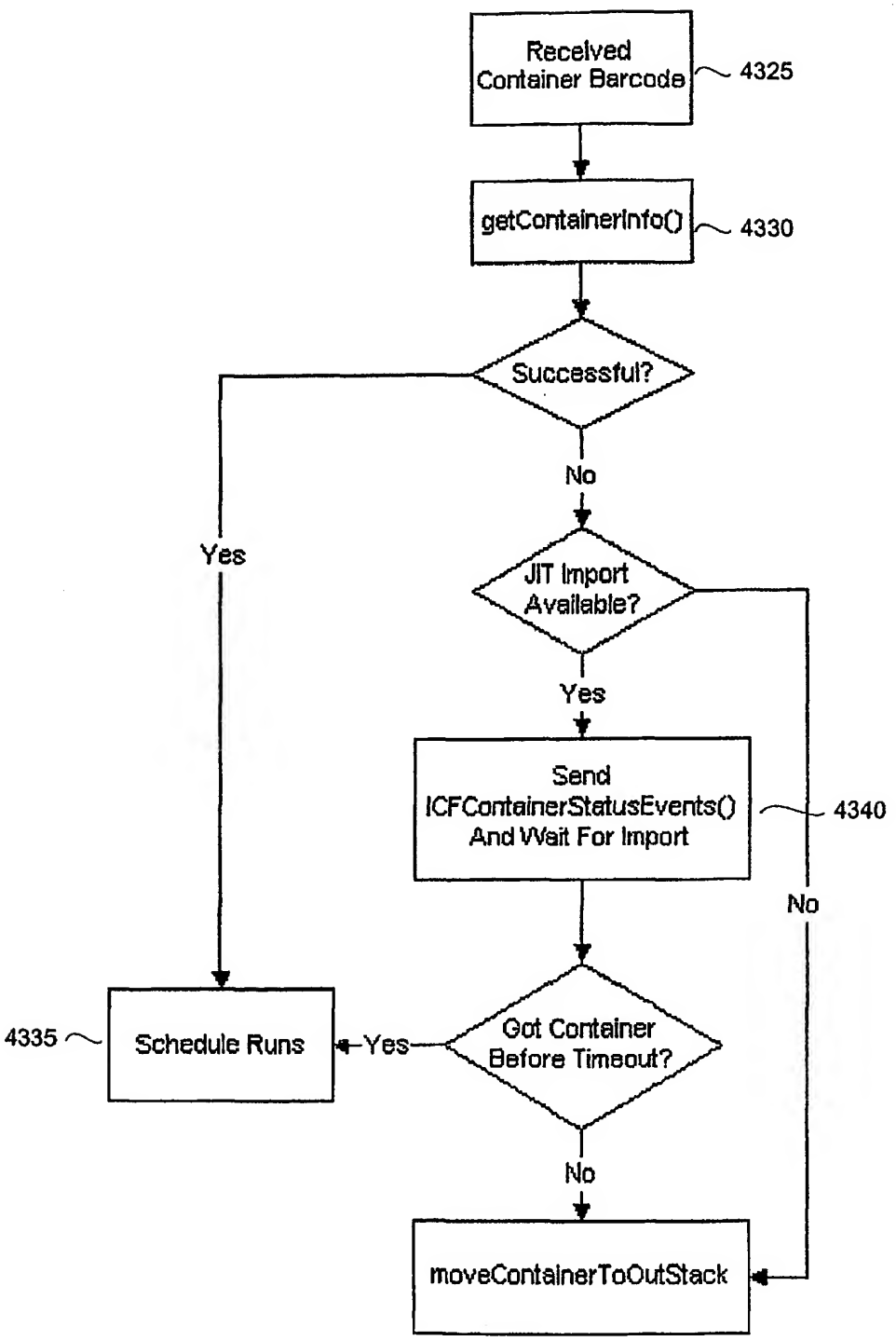

Received Container Barcode
4325
getContainerInfo()
4330
Successful?
No
Yes
JIT Import Available?
Yes
No
Send ICFContainerStatusEvents() And Wait For Import
4340
Got Container Before Timeout?
Yes
No
4335
Schedule Runs
moveContainerToOutStack

Fig. 40

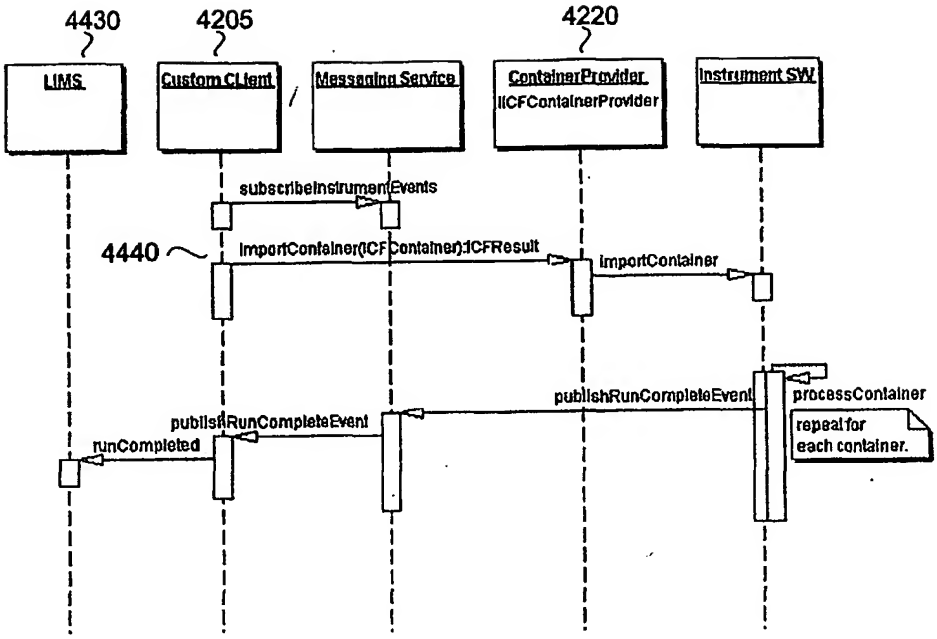

4430
4205
4220
LIMS
Custom CLient
Messaging Service
ContainerProvider
IICFContainerProvider
Instrument SW
subscribeInstrumentEvents
4440
importContainer(ICFContainer):ICFResult
importContainer
publishRunCompleteEvent
processContainer
repeat for each container.
publishRunCompleteEvent
runCompleted

Fig. 41

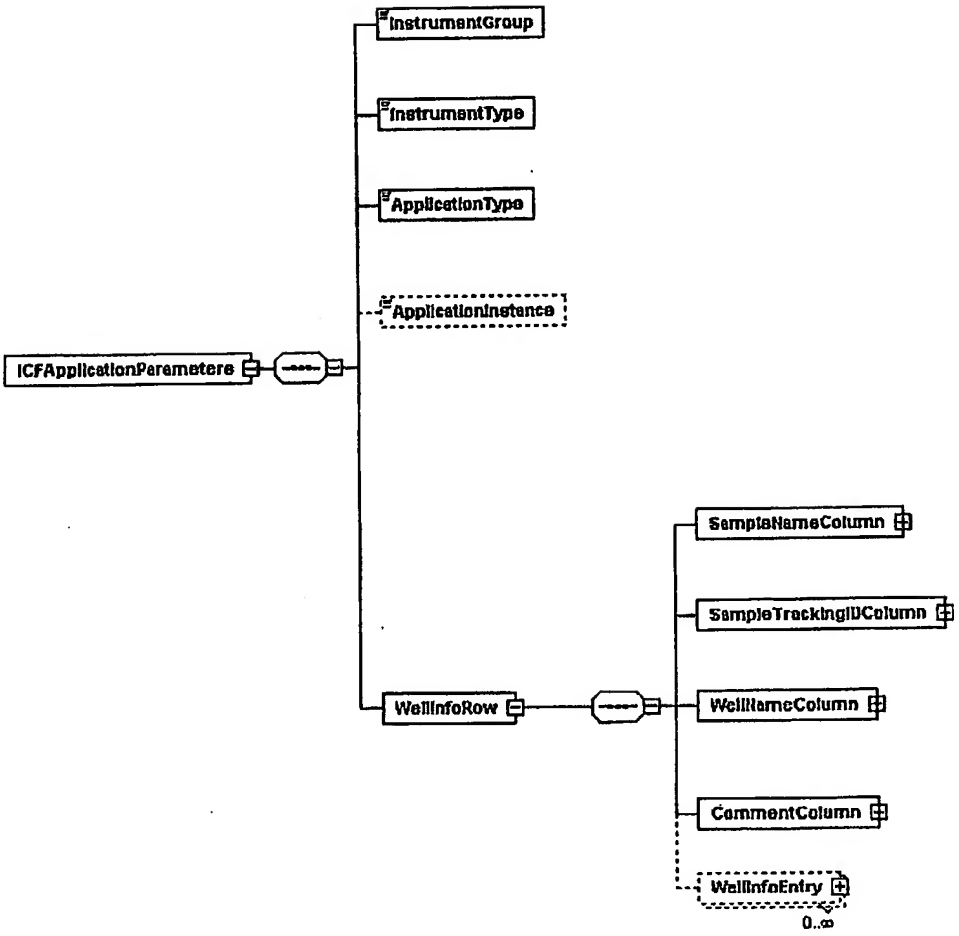

InstrumentGroup
InstrumentType
ApplicationType
ApplicationInstance
ICFApplicationParameters
SampleNameColumn
SampleTrackingIDColumn
WellInfoRow
WellNameColumn
CommentColumn
WellInfoEntry
0..∞

Fig. 42

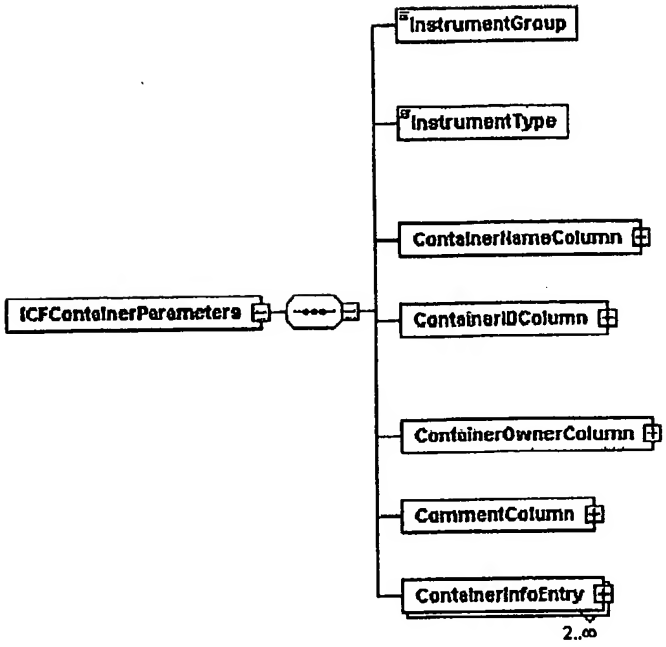

InstrumentGroup
InstrumentType
ContainerNameColumn
ICFContainerParameters
ContainerIDColumn
ContainerOwnerColumn
CommentColumn
ContainerInfoEntry
2..∞

Fig. 43

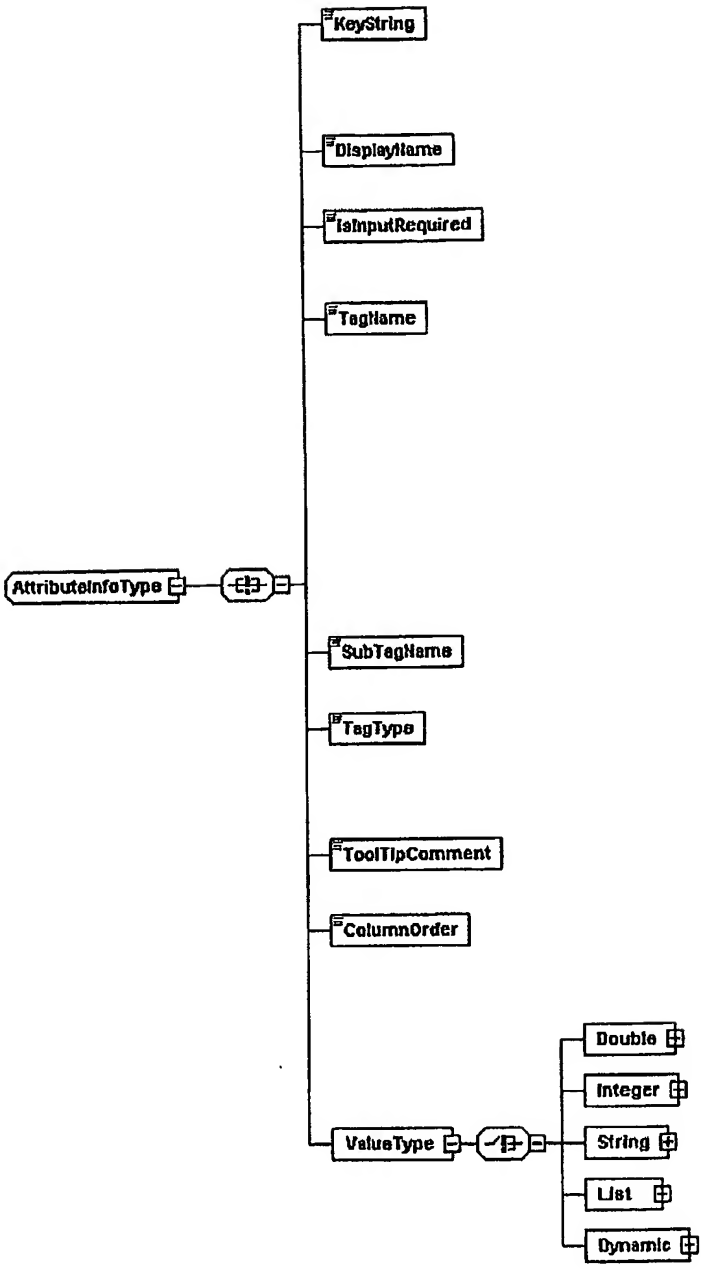

KeyString
DisplayName
IsInputRequired
TagName
AttributeInfoType
SubTagName
TagType
ToolTipComment
ColumnOrder
Double
Integer
ValueType
String
List
Dynamic

Fig. 44

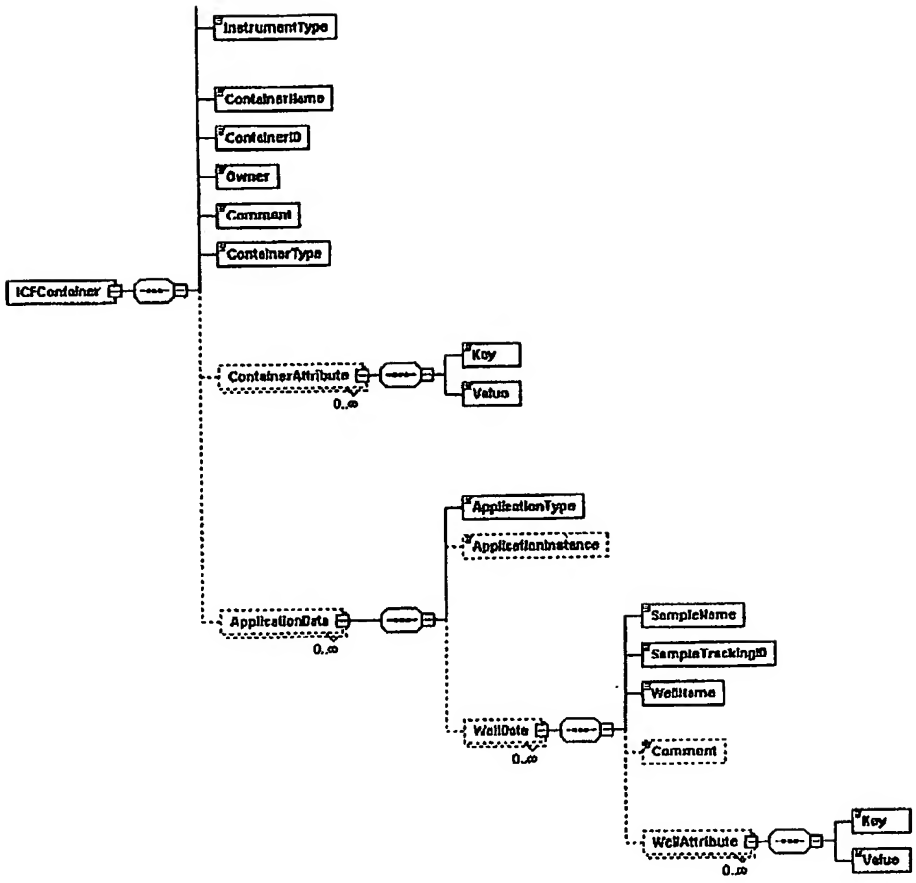

InstrumentType
ContainerName
ContainerID
Owner
Comment
ContainerType
ICFContainer
ContainerAttribute
0..∞
Key
Value
ApplicationType
ApplicationInstance
ApplicationData
0..∞
SampleName
SampleTrackingID
WellName
WellData
0..∞
Comment
WellAttribute
0..∞
Key
Value

Fig. 45

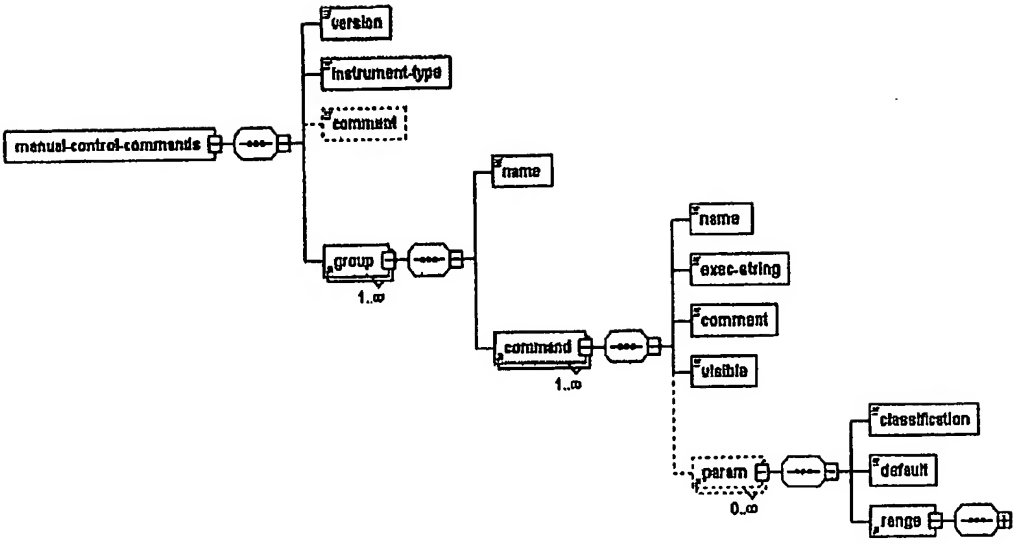

version
instrument-type
comment
manual-control-commands
name
group
1..∞
name
exec-string
comment
command
visible
1..∞
classification
default
param
range
0..∞

Fig. 46

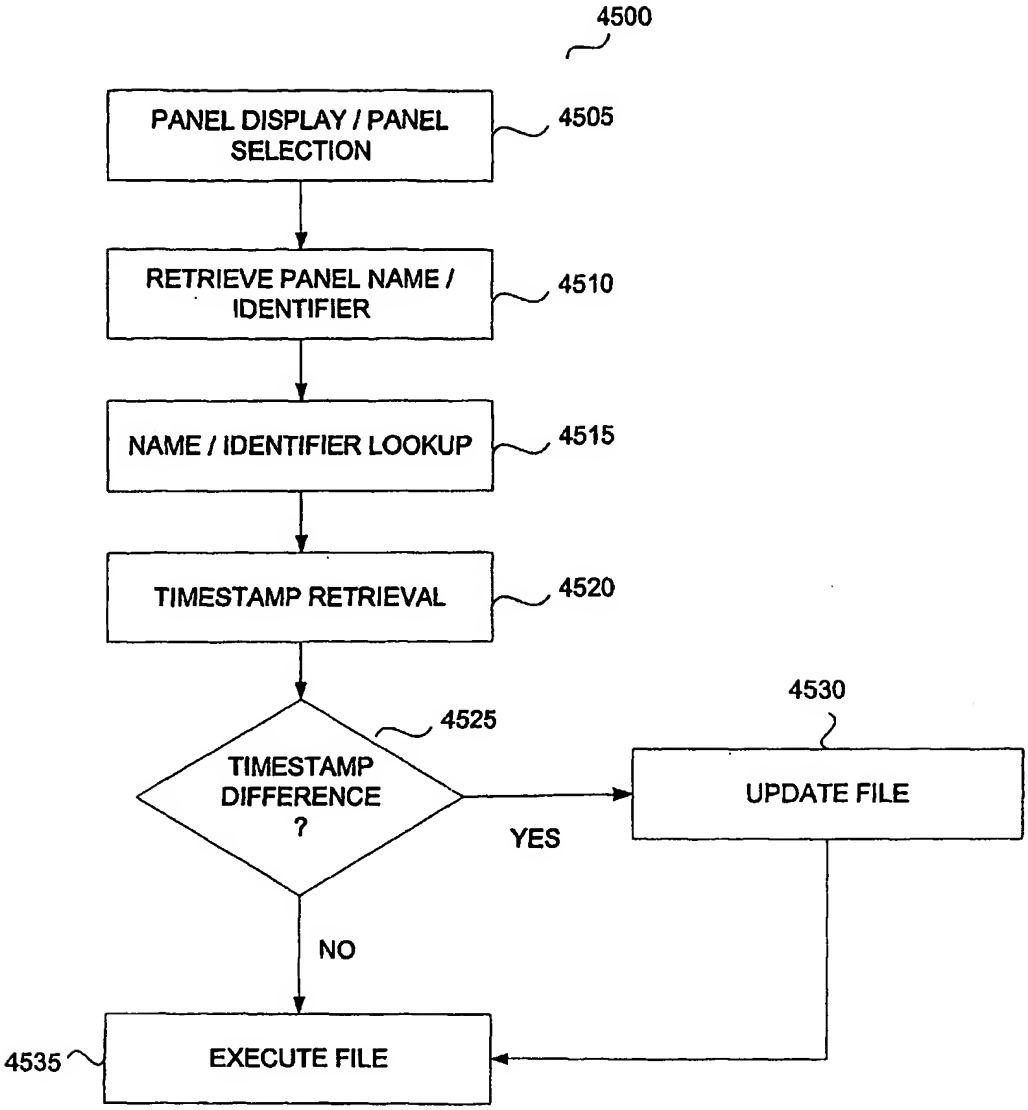

4500
PANEL DISPLAY / PANEL SELECTION
4505
RETRIEVE PANEL NAME / IDENTIFIER
4510
NAME / IDENTIFIER LOOKUP
4515
TIMESTAMP RETRIEVAL
4520
4525
TIMESTAMP DIFFERENCE ?
YES
4530
UPDATE FILE
NO
4535
EXECUTE FILE

Fig. 47